

COMP 9517 Computer Vision

Feature Representation

Outline

- Need for feature representation
- Major types of features
 - Colour
 - Texture
 - Shape
- Feature descriptors and their use in various computer vision applications

Image Features

- Image features are essentially vectors that are a compact representation of images
- They represent important information shown in an image
- Intuitive examples of image features:
 - Blobs
 - Edges
 - Corners
 - Ridges
 - Circles
 - Ellipses
 - Lines
 - Etc...



Image Features

- We need to represent images as feature vectors for further processing in a more efficient and robust way
- Examples of further processing include:
 - Object detection
 - Image segmentation
 - Image classification
 - Content-based image retrieval
 - Image stitching
 - Object tracking

Object Detection



Segmentation

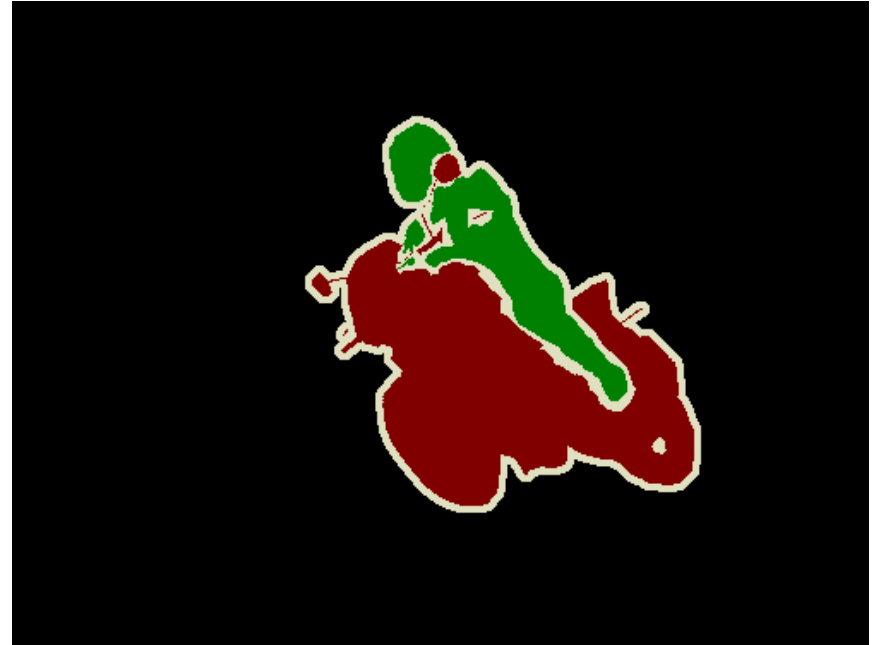
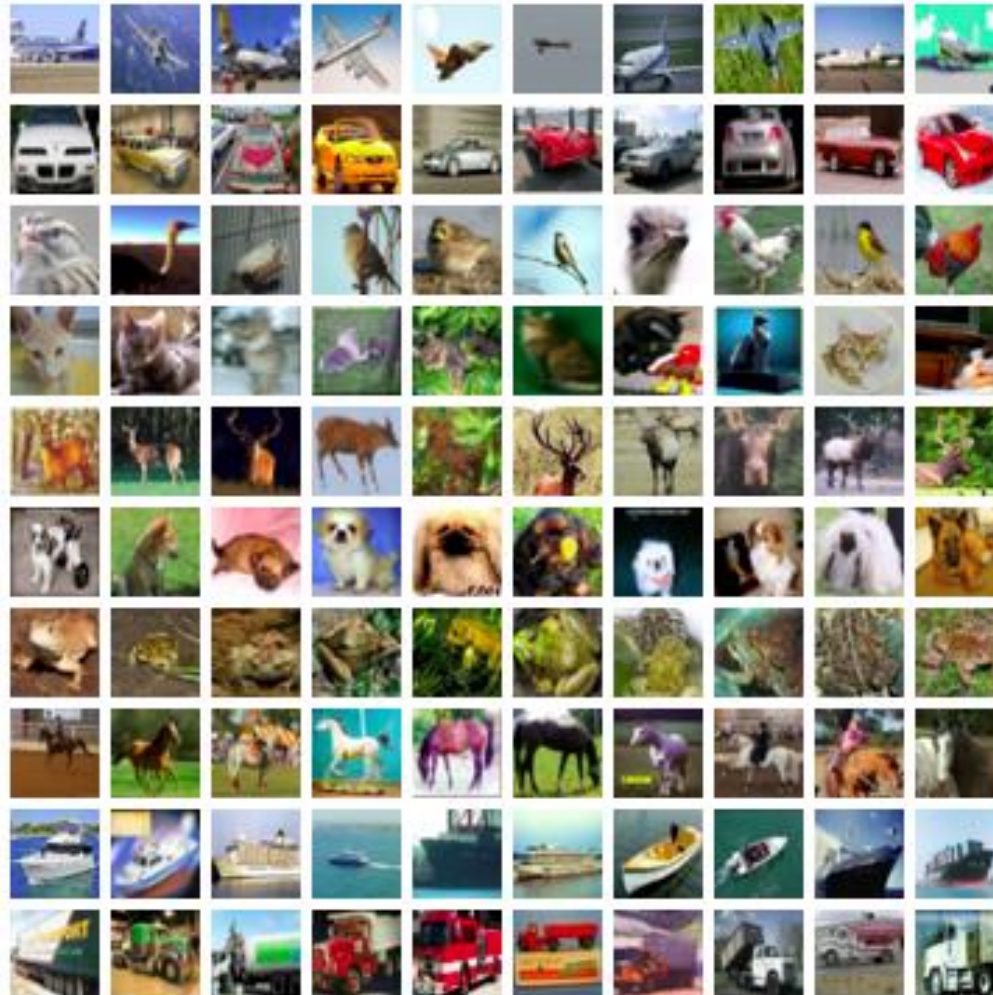


Image Classification



Content-Based Image Retrieval

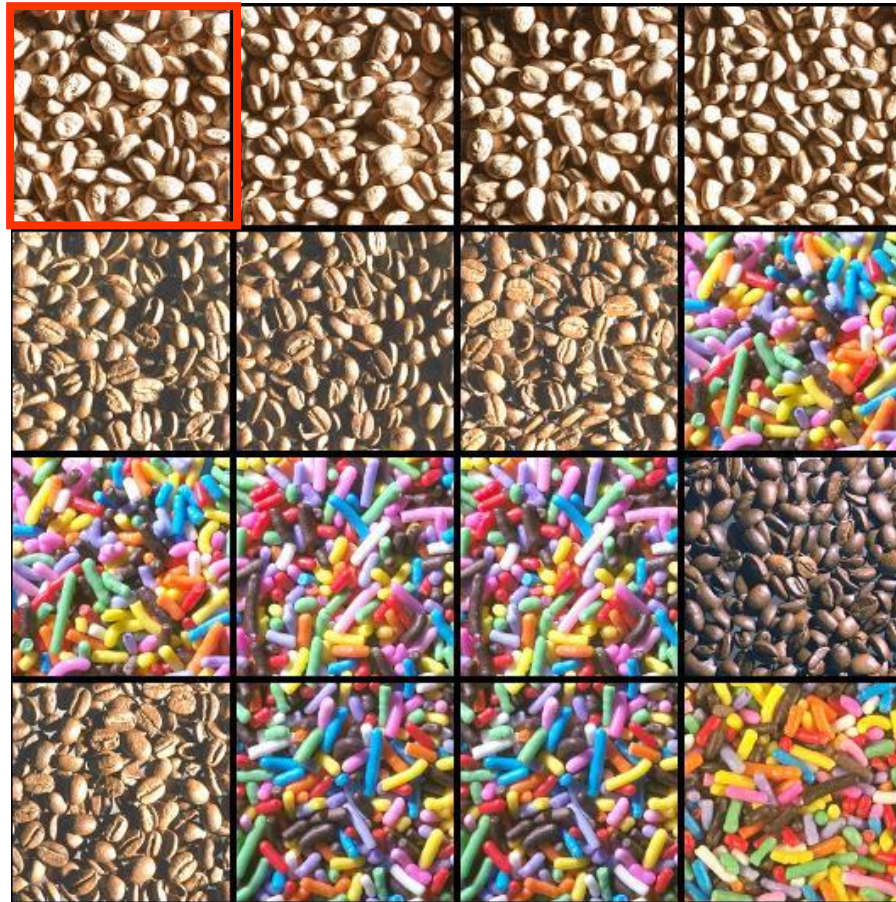
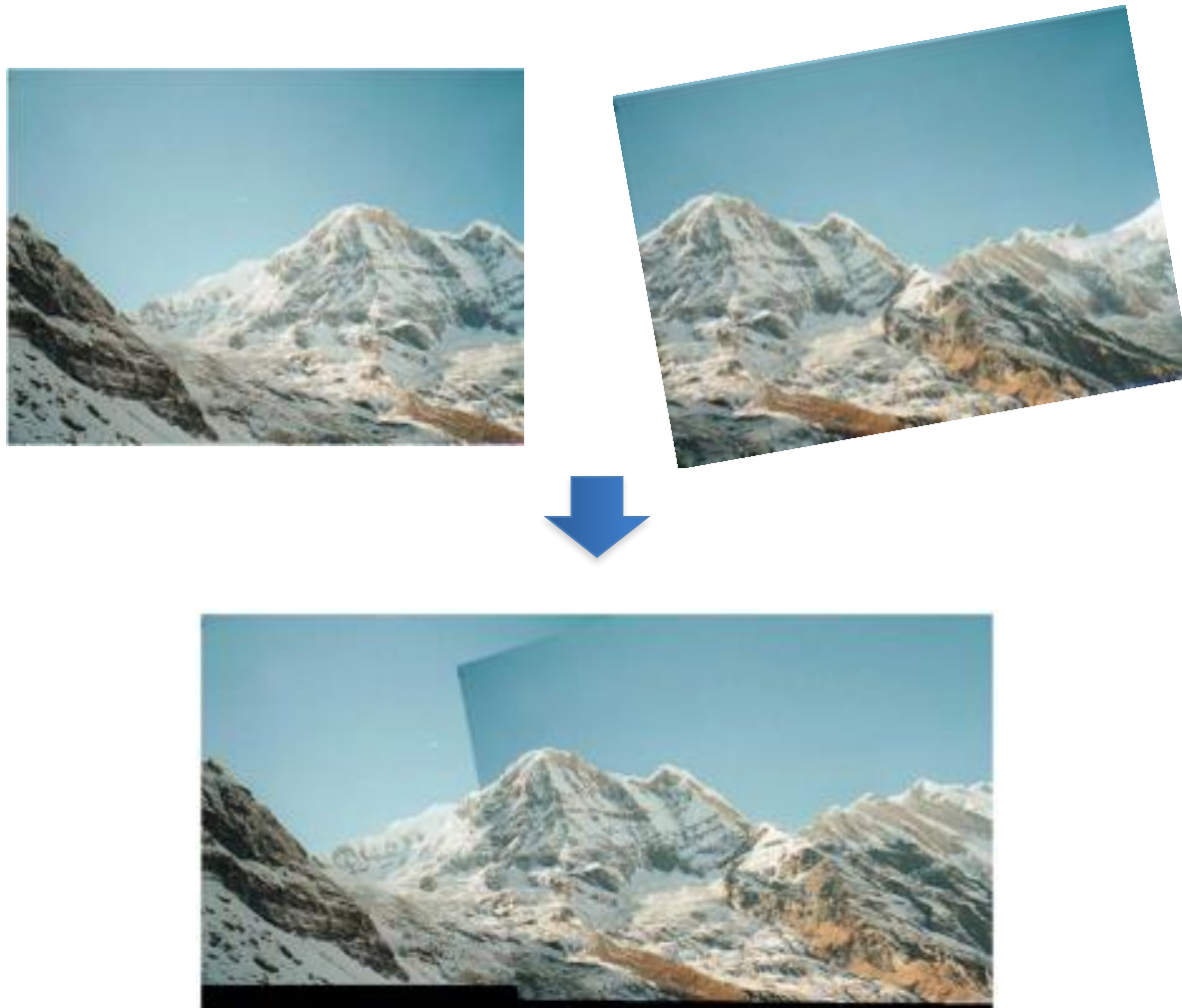
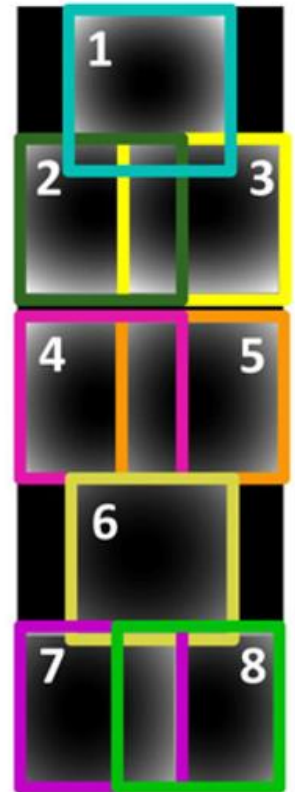
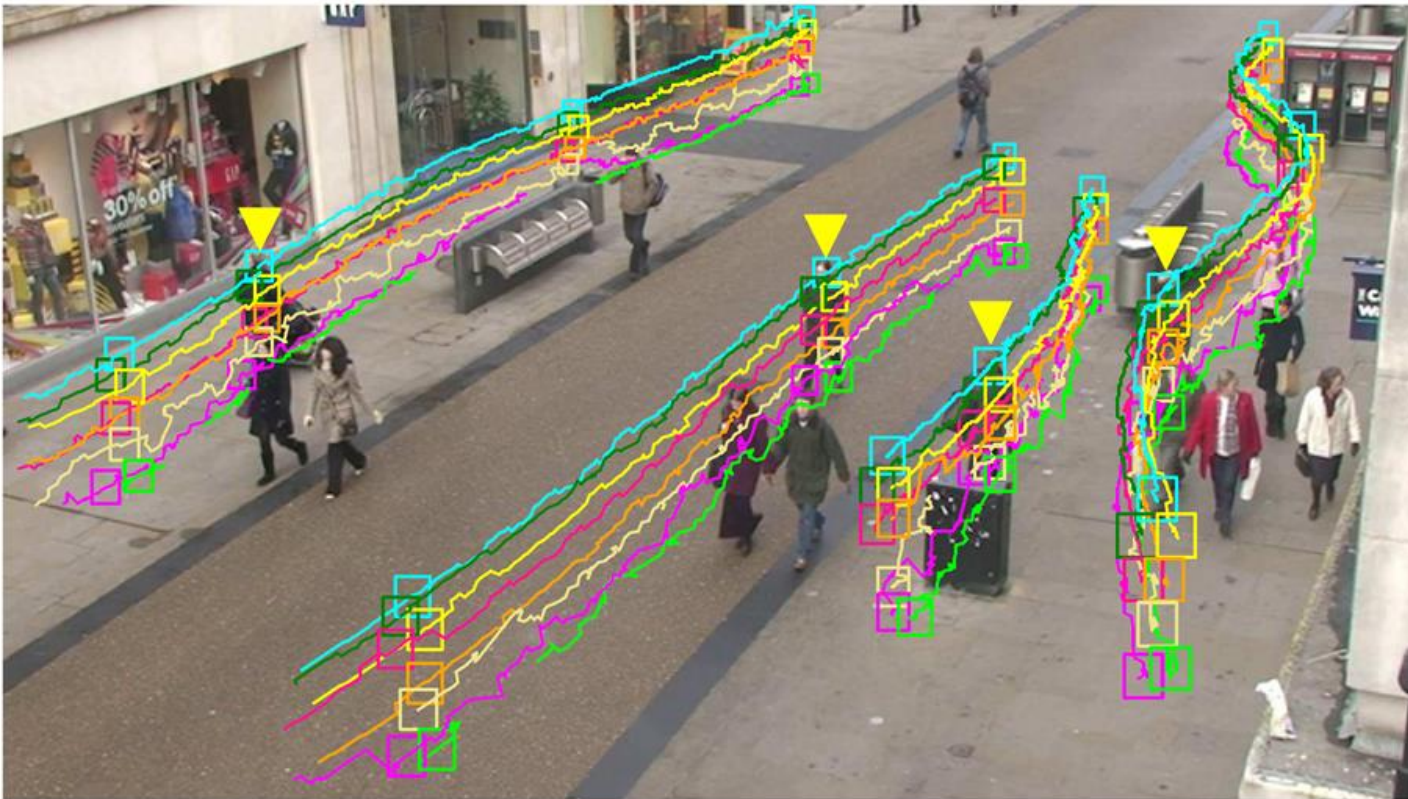


Image Stitching



Object Tracking



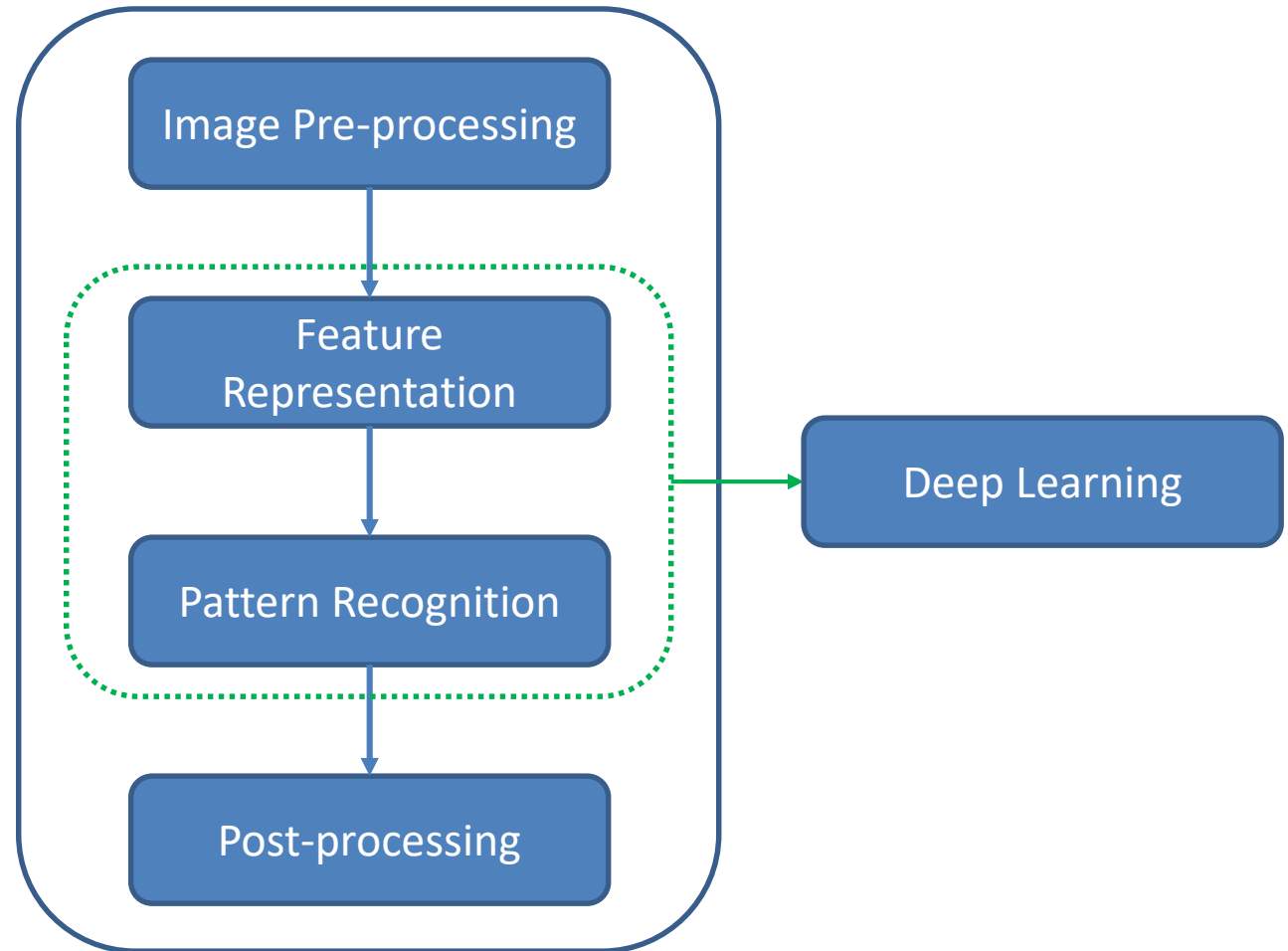
<https://heartbeat.fritz.ai/>

Properties of Features

- Why not just use pixels values directly?
 - Pixel values change with light intensity, colour and direction
 - They also change with camera orientation
 - And they are highly redundant
- Repeatability (robustness)
 - Should be detectable at the same locations in different images despite changes in illumination and viewpoint
- Saliency (descriptiveness)
 - Same points in different images should have similar features
- Compactness (efficiency)
 - Fewer features
 - Smaller features

General Framework

Object detection
Image segmentation
Image classification
Image retrieval
Image stitching
Object tracking
...



Feature Types

- Colour features
 - Colour histogram
 - Colour moments
- Texture features
 - Haralick texture features
 - Local binary patterns (LBP)
 - Scale-invariant feature transform (SIFT)
 - Texture feature encoding
- Shape features
 - Basic shape features
 - Shape context
 - Histogram of oriented gradients (HOG)

Colour Features

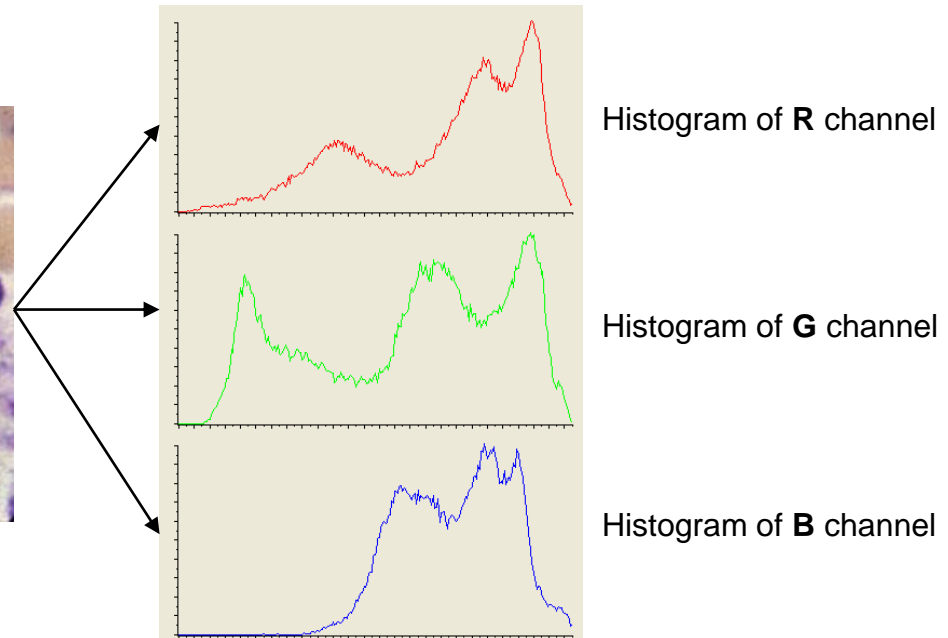
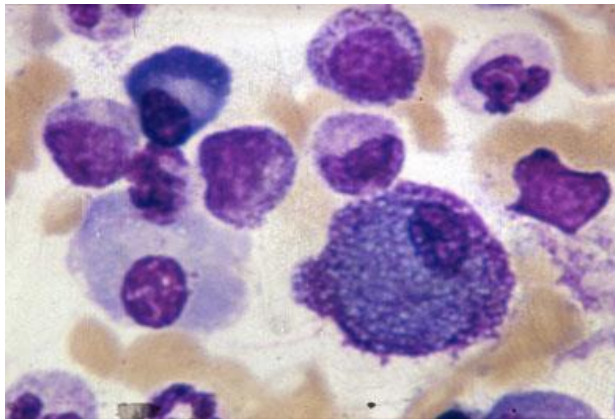
- **Colour** is the simplest feature to compute, and is invariant to image scaling, translation and rotation.
- Example: colour-based image retrieval



<http://labs.tineye.com/multicolr/>

Colour Histogram

- Represent the global distribution of pixel colours in an image
 - Step 1: Construct a histogram for each colour channel (R, G, B)
 - Step 2: Concatenate the histograms (vectors) of all channels as the final feature vector



Colour Moments

f_{ij} is the value of the i -th colour component of pixel j and N is the number of pixels in the image

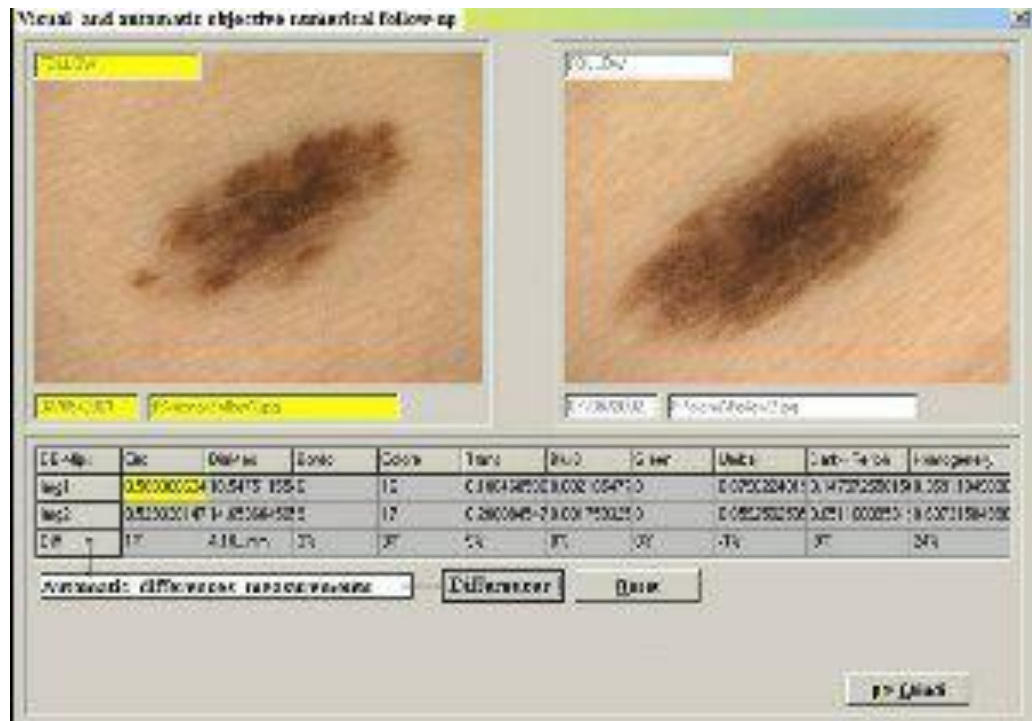
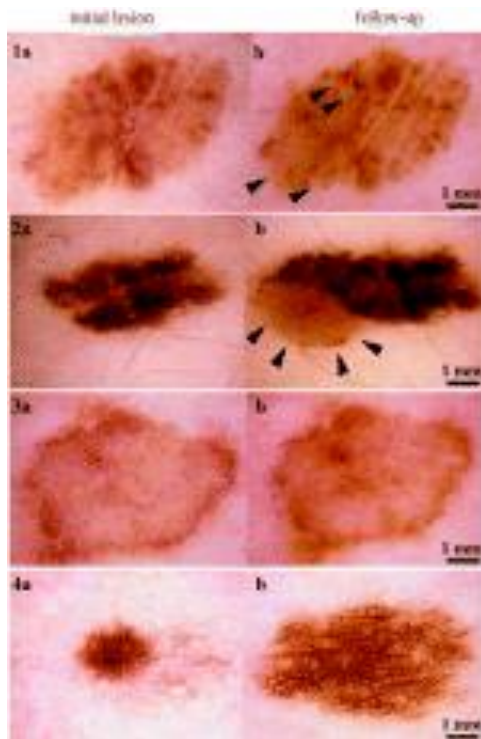
- Another way of representing colour distributions

– First-order moment	$\mu_i = \frac{1}{N} \sum_{j=1}^N f_{ij}$	(<i>mean</i>)
– Second-order moment	$\sigma_i = \left(\frac{1}{N} \sum_{j=1}^N (f_{ij} - \mu_i)^2 \right)^{\frac{1}{2}}$	(<i>variance</i>)
– Third-order moment	$s_i = \left(\frac{1}{N} \sum_{j=1}^N (f_{ij} - \mu_i)^3 \right)^{\frac{1}{3}}$	(<i>skewness</i>)

- Moments based representation of colour distributions
 - Gives a feature vector of only 9 elements (for RGB images)
 - Lower representation capability than the colour histogram

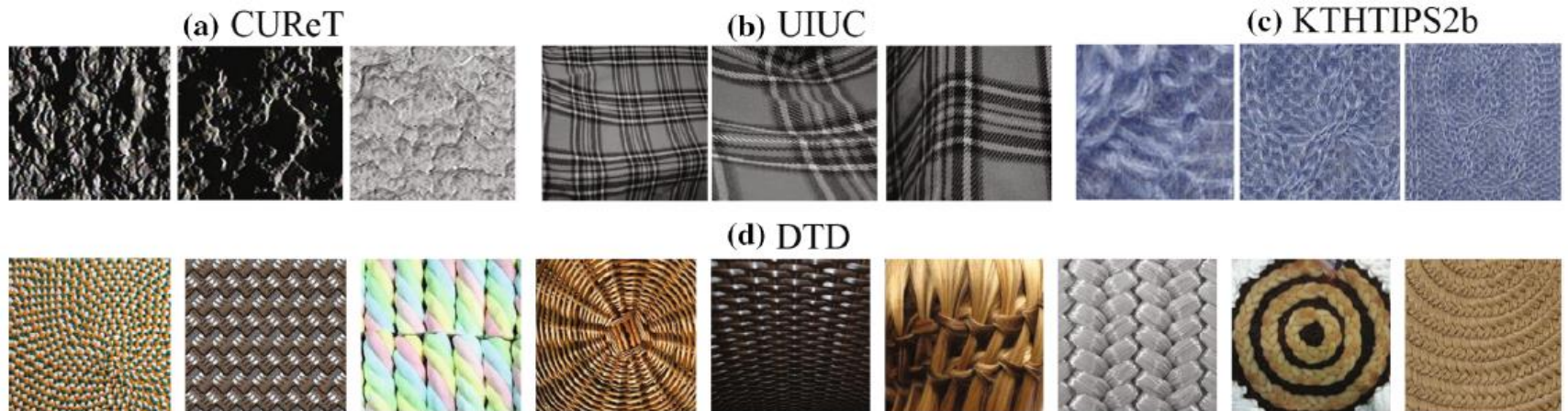
Application Example

- Colour-based image retrieval



Texture Features

- **Texture** is a powerful discriminating feature for identifying visual patterns with properties of homogeneity that cannot result from the presence of only a single color or intensity
- Example: texture classification



<https://arxiv.org/abs/1801.10324>

Haralick Features

- Haralick features give an array of statistical descriptors of image patterns to capture the spatial relationship between neighbouring pixels, that is, textures
 - Step 1: Construct the gray-level co-occurrence matrix (GLCM)
 - Step 2: Compute the Haralick feature descriptors from the GLCM

610

IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. SMC-3, NO. 6, NOVEMBER 1973

Textural Features for Image Classification

ROBERT M. HARALICK, K. SHANMUGAM, AND ITS'HAK DINSTEIN

Abstract—Texture is one of the important characteristics used in identifying objects or regions of interest in an image, whether the image be a photomicrograph, an aerial photograph, or a satellite image. This paper describes some easily computable textural features based on gray-tone spatial dependancies, and illustrates their application in category-

array. If $L_x = \{1, 2, \dots, N_x\}$ and $L_y = \{1, 2, \dots, N_y\}$ are the X and Y spatial domains, then $L_x \times L_y$ is the set of resolution cells and the digital image I is a function which assigns some gray-tone value $G \in \{1, 2, \dots, N_g\}$ to each and every

Haralick Features

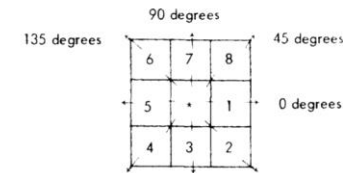


Fig. 1 Resolution cells 1 and 5 are 0° (horizontal) nearest neighbors to resolution cell *; resolution cells 2 and 6 are 135° nearest neighbors; resolution cells 3 and 7 are 90° nearest neighbors; and resolution cells 4 and 8 are 45° nearest neighbors to *. (Note this information is purely spatial, and has nothing to do with gray-tone values.)

- Step 1: Construct the GLCMs
 - Given a distance d at an orientation angle ϑ , then $p_{(d, \vartheta)}(l_1, l_2)$, being the (l_1, l_2) coefficient of the corresponding matrix $\mathbf{P}_{(d, \vartheta)}$, is the co-occurrence count or probability of going from a grey level l_1 to another grey level l_2 with an inter-sample spacing of d along the axis making an angle ϑ with the x axis.
 - If the number of distinct gray levels in the quantized image is L , then the co-occurrence matrix \mathbf{P} will be of size $L \times L$.

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

original 4×4 image

$$\mathbf{P}_{(1,0^\circ)} = \begin{bmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

$$\mathbf{P}_{(1,135^\circ)} = \begin{bmatrix} 2 & 1 & 3 & 0 \\ 1 & 2 & 1 & 0 \\ 3 & 1 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{bmatrix}$$

co-occurrence matrix construction

Haralick Features

- Step 1: Construct the GLCMs
 - For computational efficiency, the number of gray levels (L) can be reduced by binning (similar to histogram binning), e.g. $L = 256/n$, with n a constant factor.
 - Different co-occurrence matrices can be constructed by using various combinations of distance (d) and angular directions (ϑ).
 - On their own, these co-occurrence matrices do not provide any measure of texture that can easily be used as texture descriptors.
 - The information in the co-occurrence matrices needs to be further extracted as a set of feature values => Haralick descriptors.

Haralick Features

- Step 2: Compute the Haralick descriptors from the GLCMs
 - One set of Haralick descriptors for each GLCM corresponding to a particular distance (d) and angular direction (ϑ)

Angular Second Moment

$$\sum_i \sum_j p(i, j)^2$$

Contrast

$$\sum_{n=0}^{N_g-1} n^2 \{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j) \}, |i - j| = n$$

Correlation

$$\frac{\sum_i \sum_j (ij) p(i, j) - \mu_x \mu_y}{\sigma_x \sigma_y}$$

where μ_x , μ_y , σ_x , and σ_y
are the means and std. deviations
of p_x and p_y , the partial probability
density functions

Sum of Squares: Variance

$$\sum_i \sum_j (i - \mu)^2 p(i, j)$$

Inverse Difference Moment

$$\sum_i \sum_j \frac{1}{1 + (i - j)^2} p(i, j)$$

Sum Average

$$\sum_{i=2}^{2N_g} i p_{x+y}(i)$$

where x and y are the coordinates (row and
column) of an entry in the co-occurrence matrix,
and $p_{x+y}(i)$ is the probability of co-occurrence
matrix coordinates summing to $x + y$

Haralick Features

- Step 2: Compute the Haralick descriptors from the GLCMs
 - One set of Haralick descriptors for each GLCM corresponding to a particular distance (d) and angular direction (ϑ)

Sum Variance	$\sum_{i=2}^{2N_g} (i - f_s)^2 p_{x+y}(i)$
Sum Entropy	$-\sum_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\} = f_s$
Entropy	$-\sum_i \sum_j p(i, j) \log(p(i, j))$
Difference Variance	$\sum_{i=0}^{N_g-1} i^2 p_{x-y}(i)$
Difference Entropy	$-\sum_{i=0}^{N_g-1} p_{x-y}(i) \log\{p_{x-y}(i)\}$
Info. Measure of Correlation 1	$\frac{HXY - HXY1}{\max\{HX, HY\}}$
Info. Measure of Correlation 2	$(1 - \exp[-2(HXY2 - HXY)])^{\frac{1}{2}}$ where $HXY = -\sum_i \sum_j p(i, j) \log(p(i, j))$, HX , HY are the entropies of p_x and p_y , $HXY1 =$ $-\sum_i \sum_j p(i, j) \log\{p_x(i)p_y(j)\}$ $HXY2 =$ $-\sum_i \sum_j p_x(i)p_y(j) \log\{p_x(i)p_y(j)\}$
Max. Correlation Coeff.	Square root of the second largest eigenvalue of \mathbf{Q} where $\mathbf{Q}(i, j) = \sum_k \frac{p(i, k)p(j, k)}{p_x(i)p_y(k)}$

Application Example

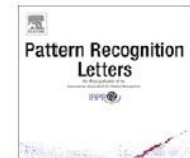
Pattern Recognition Letters 29 (2008) 1684–1693

<https://doi.org/10.1016/j.patrec.2008.04.013>



Contents lists available at ScienceDirect

Pattern Recognition Letters



WND-CHARM: Multi-purpose transforms

Nikita Orlov^a, Lior Shamir^{a,*}, Tomasz M

^a Image Informatics and Computational Biology Unit, Laboratory of

^b Computer Laboratory, University of Cambridge, 15 Thomson Ave

ARTICLE INFO

Article history:

Received 27 June 2007

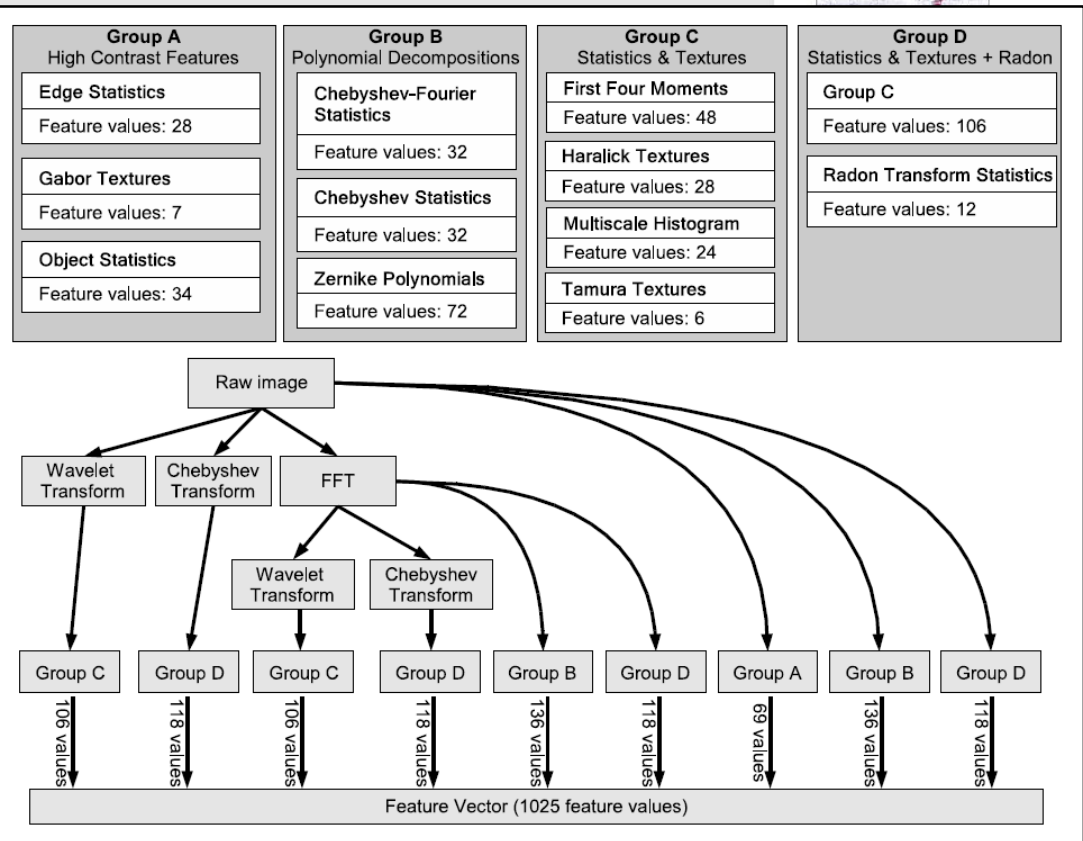
Received in revised form 4 April 2008

Available online 7 May 2008

Communicated by M.-J. Li

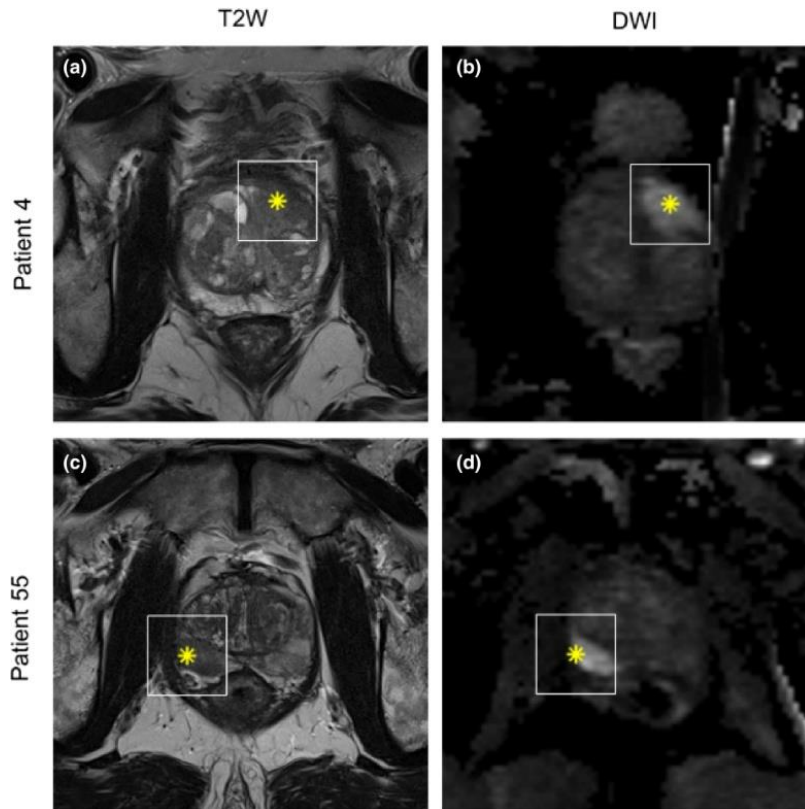
ABSTRACT

We describe tasks with off-the-shelf image features. These features are the image



Application Example

- Commonly used nowadays in medical imaging studies due to its simplicity and interpretability



C. Jensen et al.

Assessment of prostate cancer prognostic Gleason grade group using zonal-specific features extracted from biparametric MRI using a KNN classifier

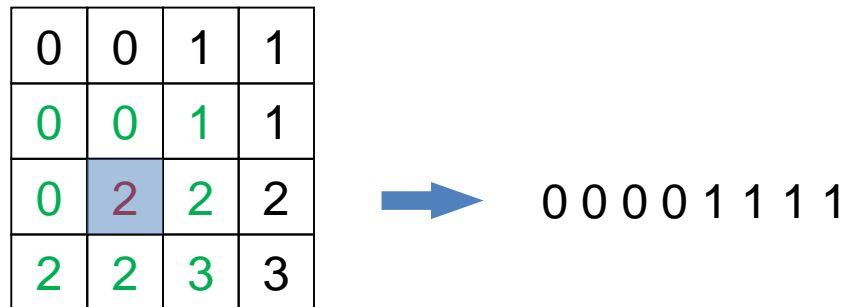
Journal of Applied Clinical Medical Physics, 2019

<https://doi.org/10.1002/acm2.12542>

1. Pre-processing
2. Extract Haralick, run-length, and histogram features from the region of interest
3. Feature selection
4. Classification using kNN

Local Binary Patterns

- Describe the spatial structure of local image texture
 - Divide the image into cells of $N \times N$ pixels (e.g. $N = 16$ or 32)
 - Compare each pixel in a cell to each of its 8 neighbouring pixels:
If the centre pixel's value is greater than the neighbour's value, write "0", otherwise write "1"
 - This gives an 8-digit binary pattern per pixel after comparing with all 8 neighbouring pixels, representing a value in the range 0...255



Local Binary Patterns

- Describe the spatial structure of local image texture (cont.)
 - Generate the histogram for all pixels in the cell, computing the frequency of each 8-digit binary number occurring in the cell
 - This gives a 256-bin histogram (the LBP feature vector)
 - Combine the histograms of all cells to obtain the image-level LBP feature descriptor

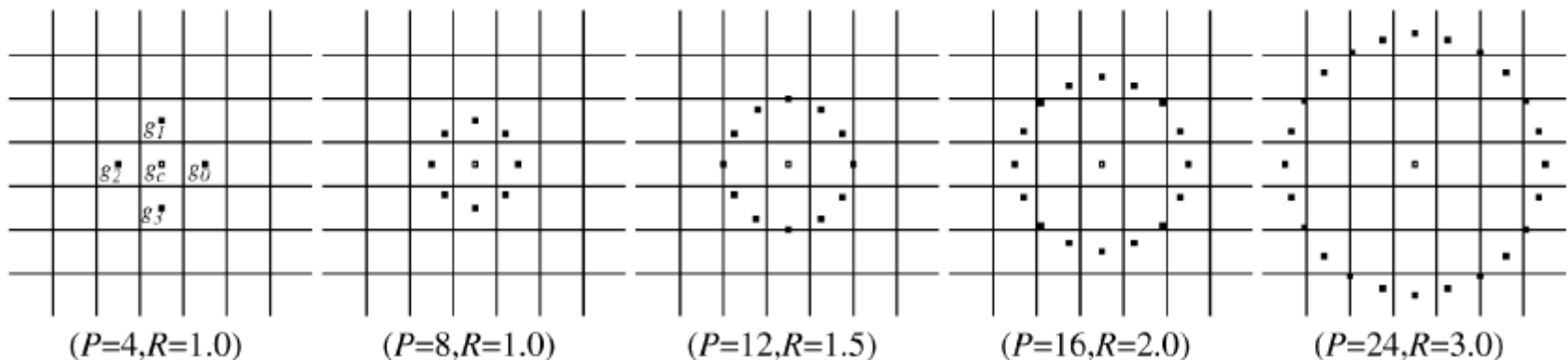
0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3



A histogram of 256 elements

Local Binary Patterns

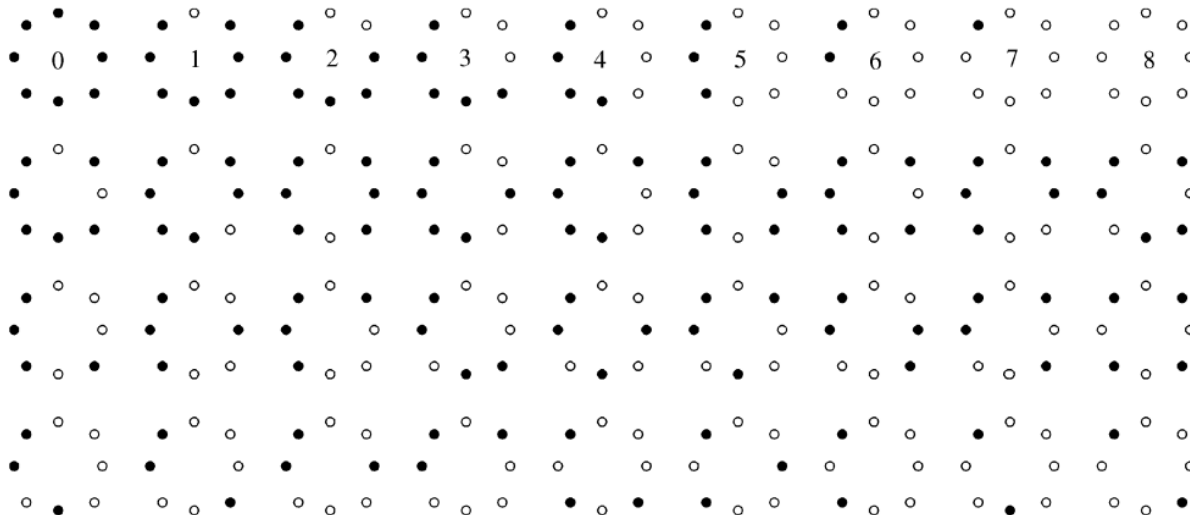
- LBP can be multi-resolution and rotation-invariant
 - Multi-resolution: varying the distance between the centre pixel and neighbouring pixels, and the number of neighbouring pixels



T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):971-987, 2002. <https://doi.org/10.1109/TPAMI.2002.1017623>

Local Binary Patterns

- LBP can be multi-resolution and rotation-invariant
 - Rotation-invariant: varying the way of constructing the 8-digit binary number, e.g. performing bitwise shift to derive the smallest number => this reduces the LBP feature dimension from 256 to 36

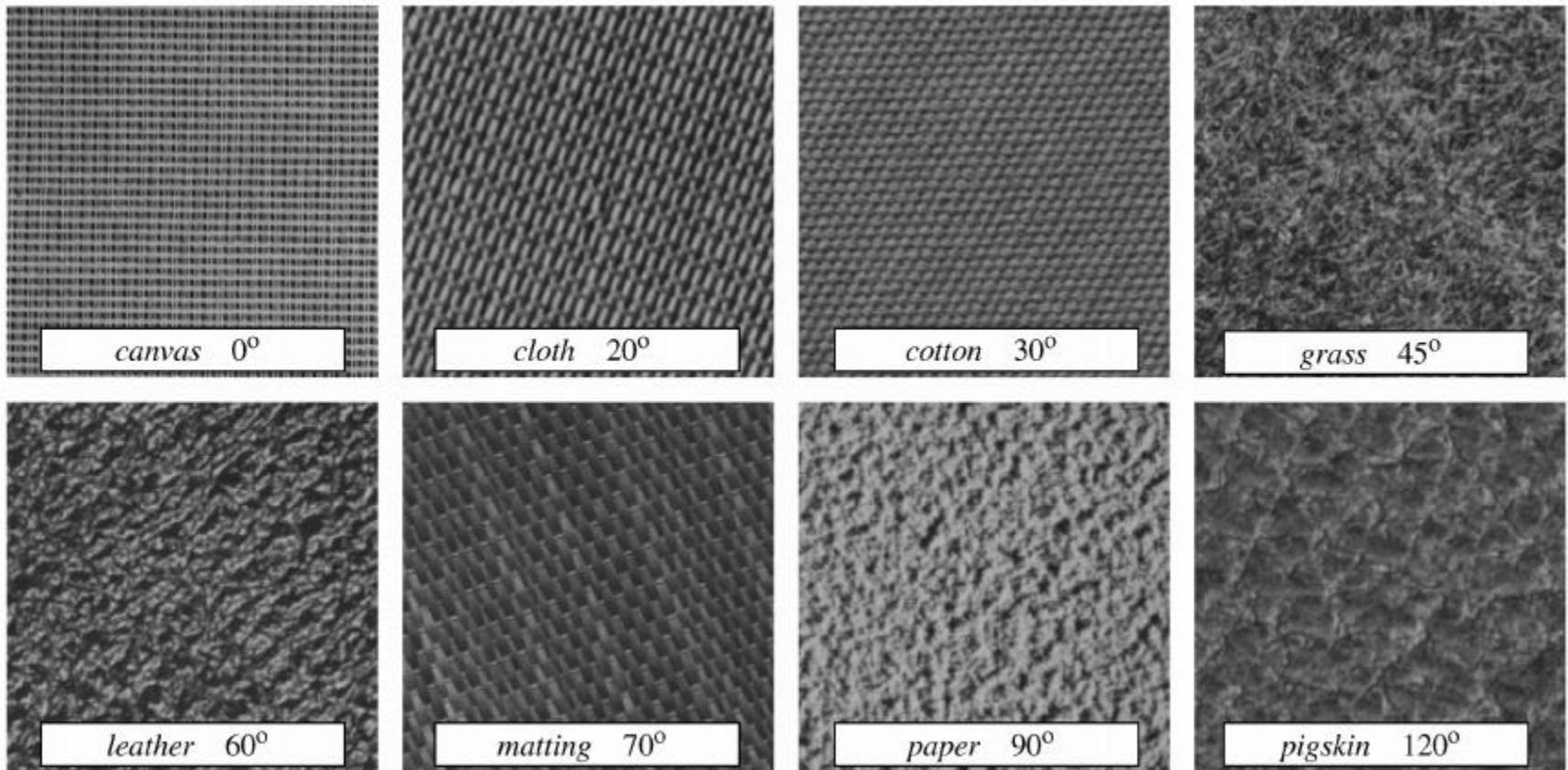


T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):971-987, 2002. <https://doi.org/10.1109/TPAMI.2002.1017623>

Application Example

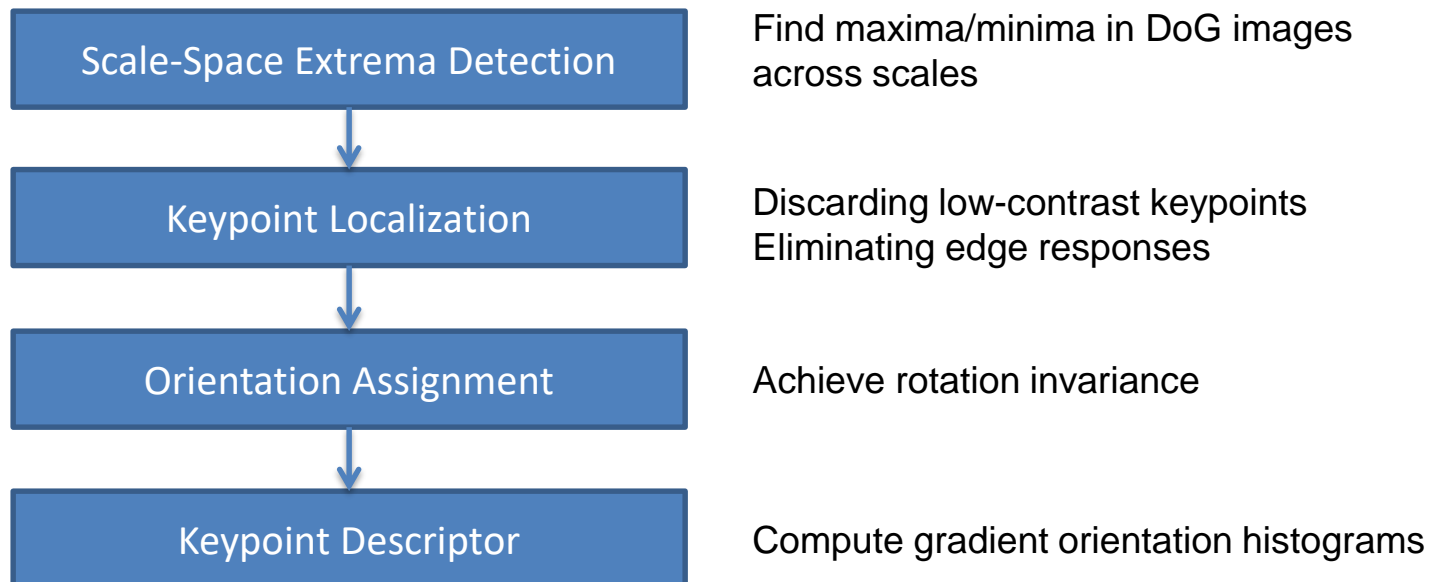
- Texture classification

P,R	$LBP_{P,R}$	
	BINS	RESULT
8,1	10	88.2
16,2	18	98.5
24,3	26	99.1
8,1+16,2	10+18	99.0
8,1+24,3	10+26	99.6
16,2+24,3	18+26	99.0
8,1+16,2+24,3	10+18+26	99.1



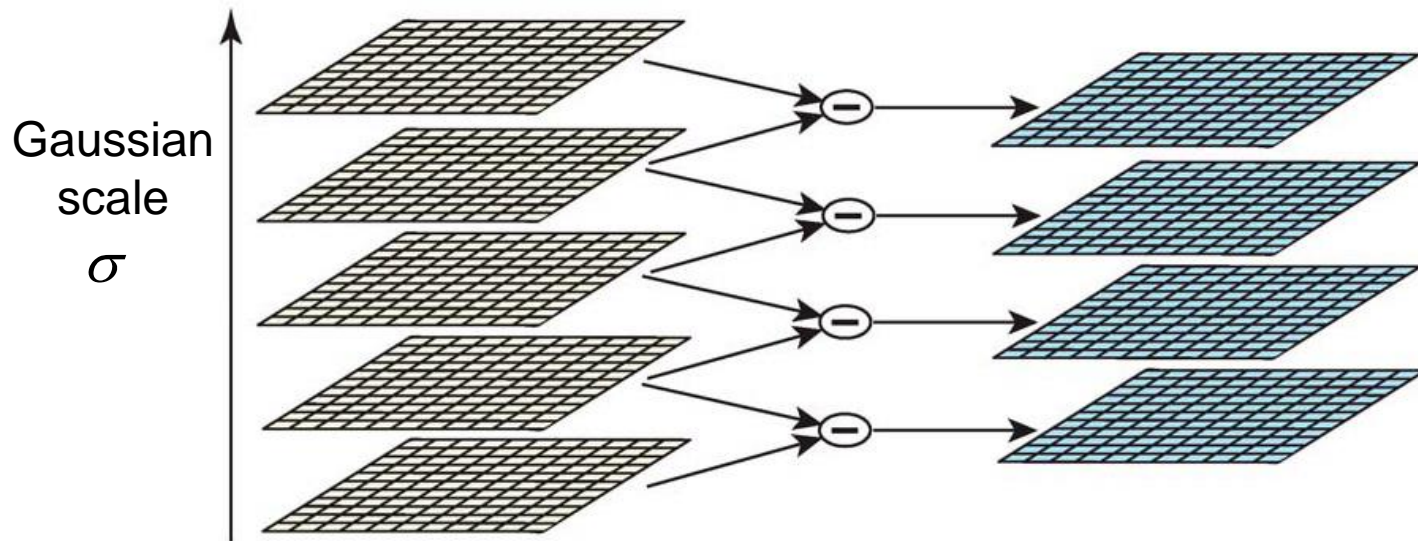
Scale-Invariant Feature Transform

- SIFT feature describes the texture features in a localised region around a keypoint
- SIFT descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes



SIFT Extrema Detection

- Detect maxima and minima in the scale space of the image

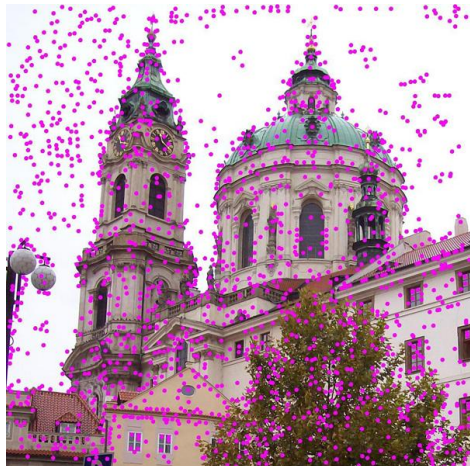


$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis. 60(2):91-110, November 2004. <https://doi.org/10.1023/B:VISI.0000029664.99615.94>

SIFT Keypoint Localization

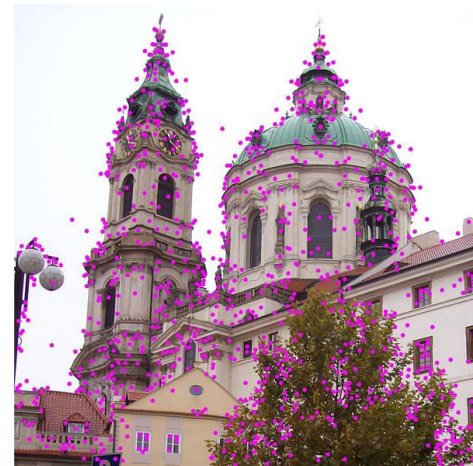
- Improve and reduce the set of found keypoints
 - Use 3D quadratic fitting in scale-space to get subpixel optima
 - Reject low-contrast and edge points using Hessian analysis



Initial keypoints from
scale-space optima



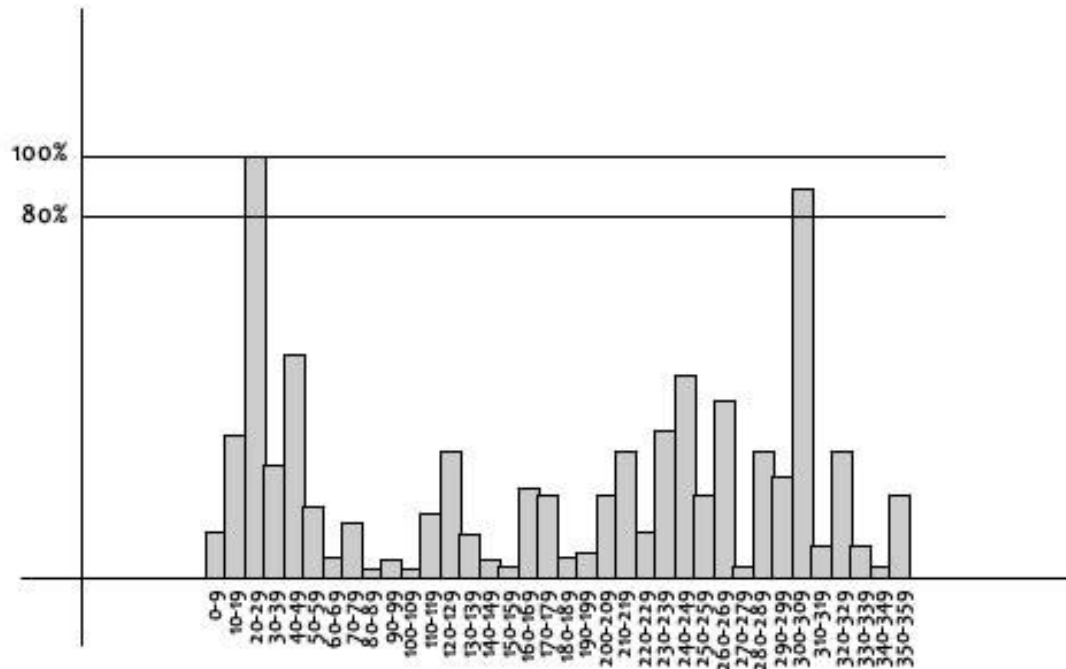
Keypoints after rejecting
low-contrast points



Final keypoints after
rejecting edge points

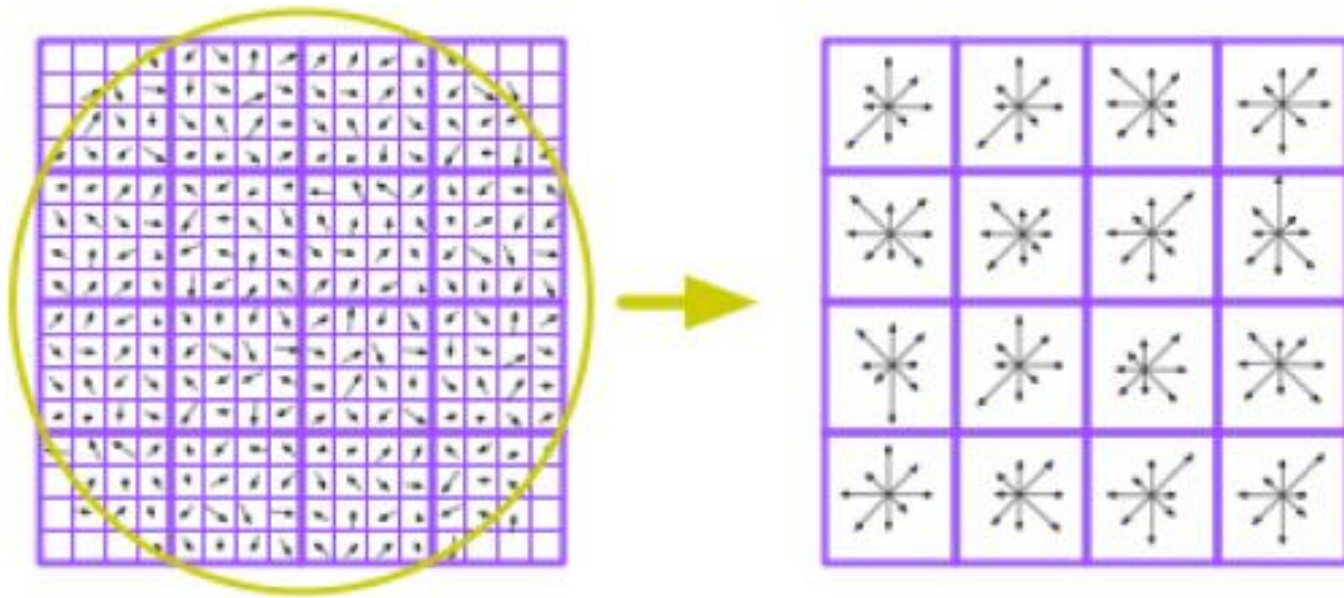
SIFT Orientation Assignment

- Estimate keypoint orientation using local gradient vectors
 - Make an orientation histogram of local gradient vectors
 - Find the dominant orientations from the peaks of the histogram



SIFT Keypoint Descriptor

- 4 x 4 array of gradient histogram weighted by magnitude
- 8 bins in gradient orientation histogram
- Total 8 x 4 x 4 array = 128 dimensions



Application Example

- Image matching



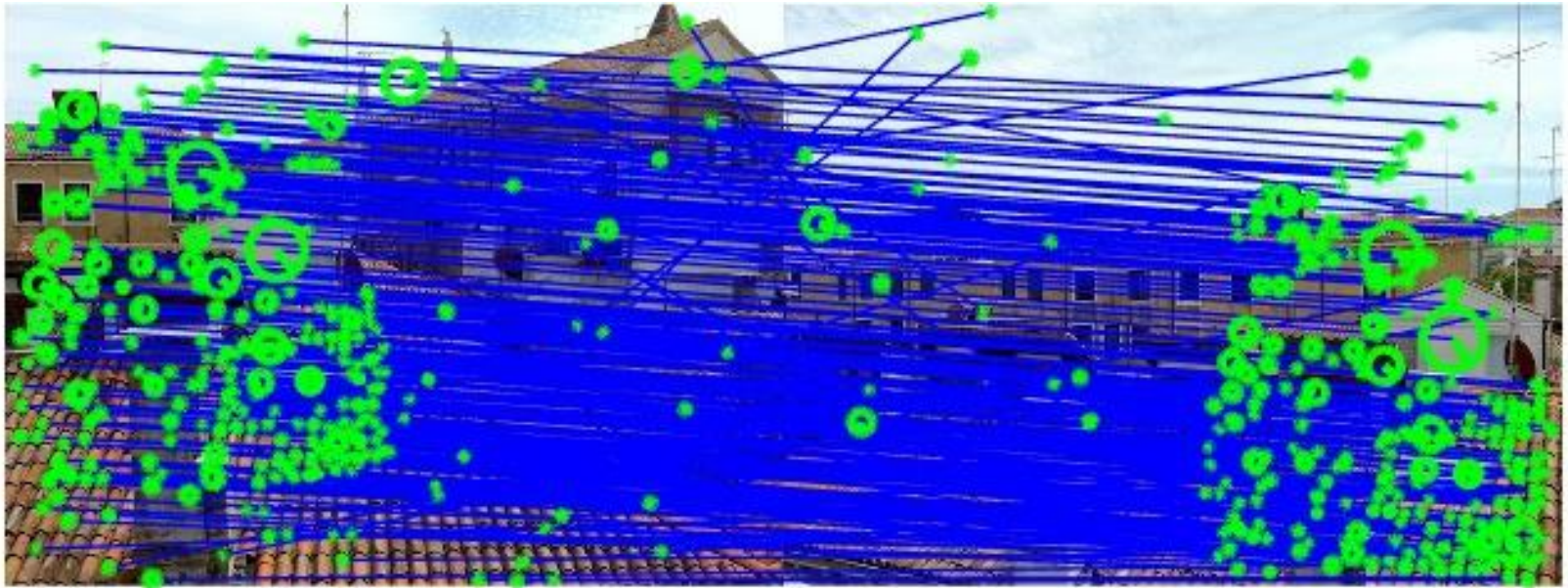
Application Example

- Image matching
 - Find SIFT keypoints



Application Example

- Image matching
 - Find best matching between SIFT keypoints

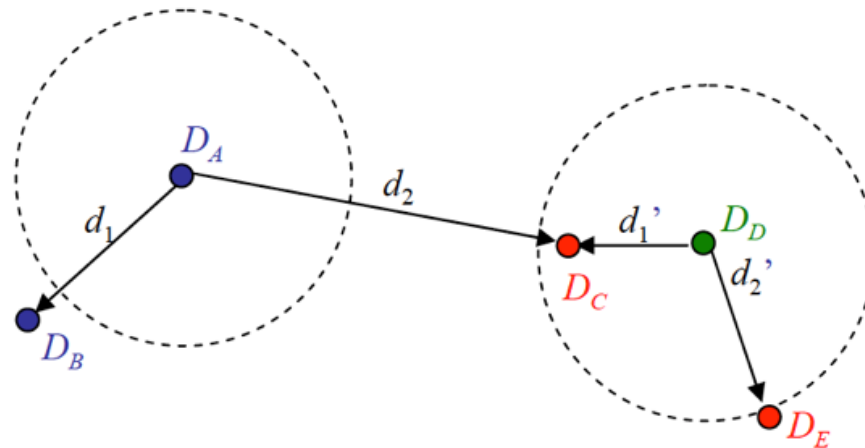


Descriptor Matching

- Nearest Neighbour Distance Ratio

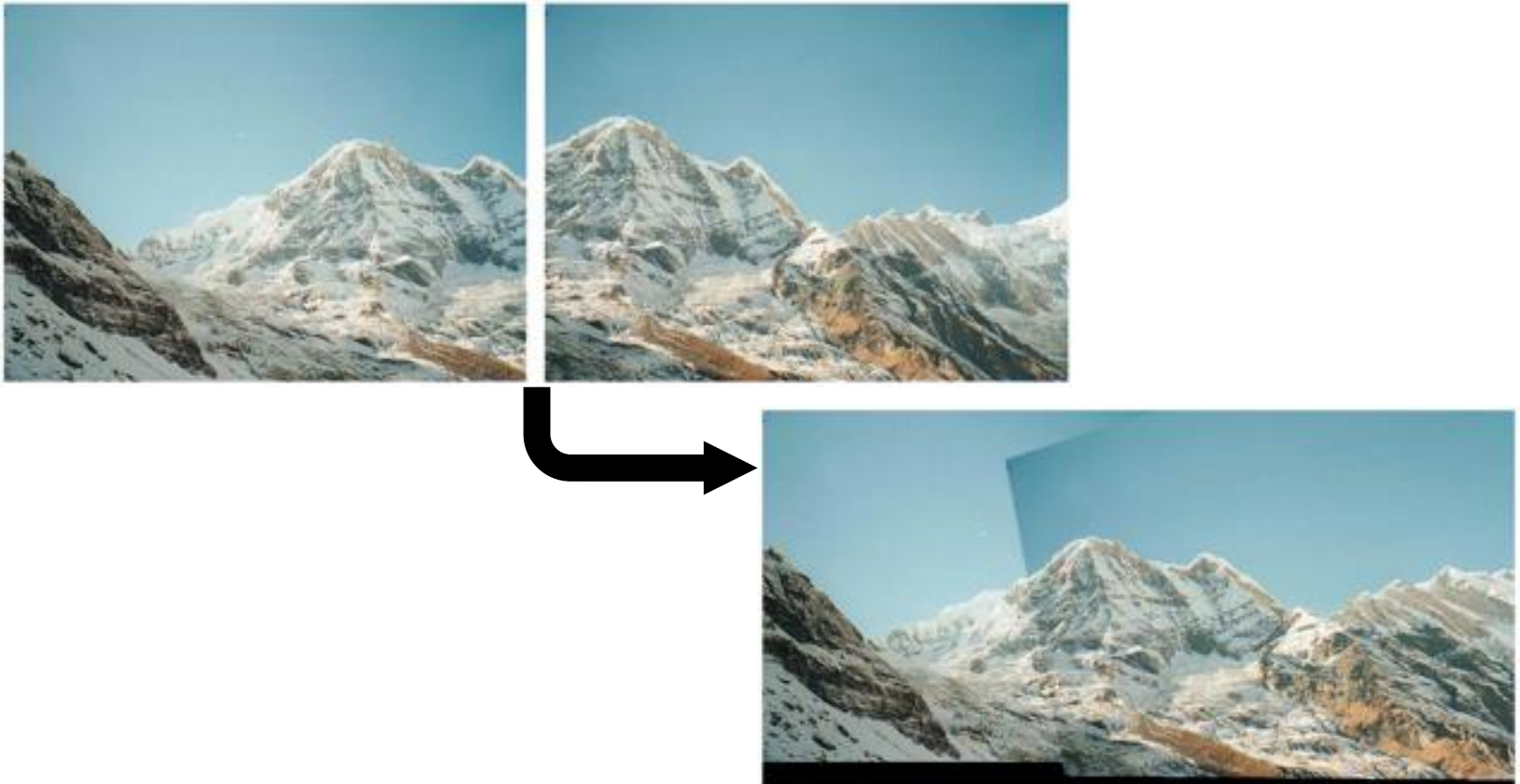
$$NNDR = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|}$$

- d_1 is the distance to the first nearest neighbour
- d_2 is the distance to the second nearest neighbour
- Neighbours in feature space



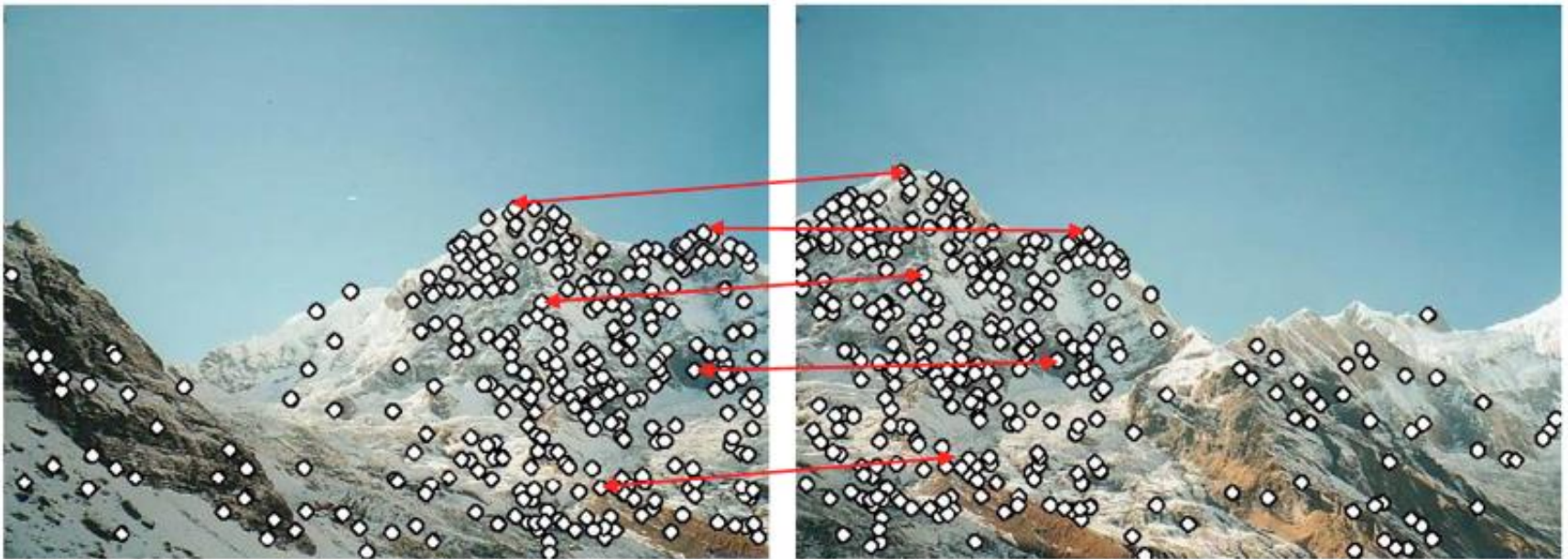
Application Example

- Image stitching



Application Example

- Image stitching
 - Find SIFT keypoints and feature correspondences



Application Example

- Image stitching
 - Find the right spatial transformation



Transformations



original



translation



rotation



scale



affine



perspective

Transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Projective

Fitting and Alignment

- Least-squares (LS) fitting of corresponding keypoints $(\mathbf{x}_i, \mathbf{x}'_i)$

$$E_{LS} = \sum_i \|\mathbf{r}_i\|^2 = \sum_i \|f(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i\|^2$$

where \mathbf{p} are the parameters of the transformation f

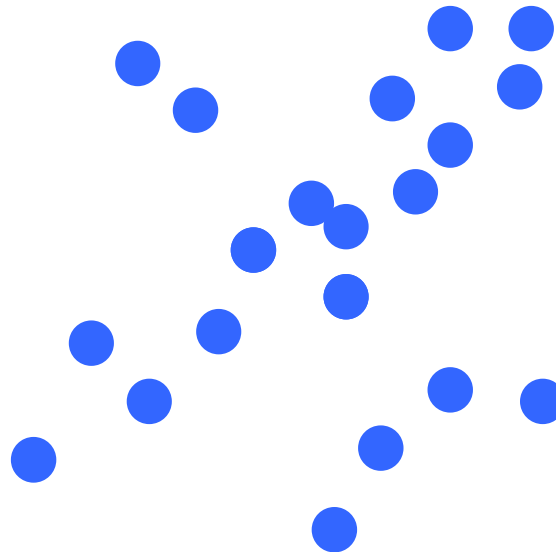
$$\begin{aligned} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} &= \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \dots & & & \\ & & \dots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ d \\ e \\ c \\ f \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ \vdots \end{bmatrix} \\ & \quad \Downarrow \\ \mathbf{p} &= [\mathbf{A}^T \mathbf{A}]^{-1} \mathbf{A}^T \mathbf{b} \quad \Leftarrow \quad \mathbf{A} \mathbf{p} = \mathbf{b} \end{aligned}$$

Fitting and Alignment

- RANdom SAmple Consensus (RANSAC) fitting
 - Least-squares fitting is hampered by outliers
 - Some kind of outlier detection and rejection is needed
 - Better use a subset of the data and check inlier agreement
 - RANSAC does this in a iterative way to find the optimum

Fitting and Alignment

- RANSAC



(line fitting example)

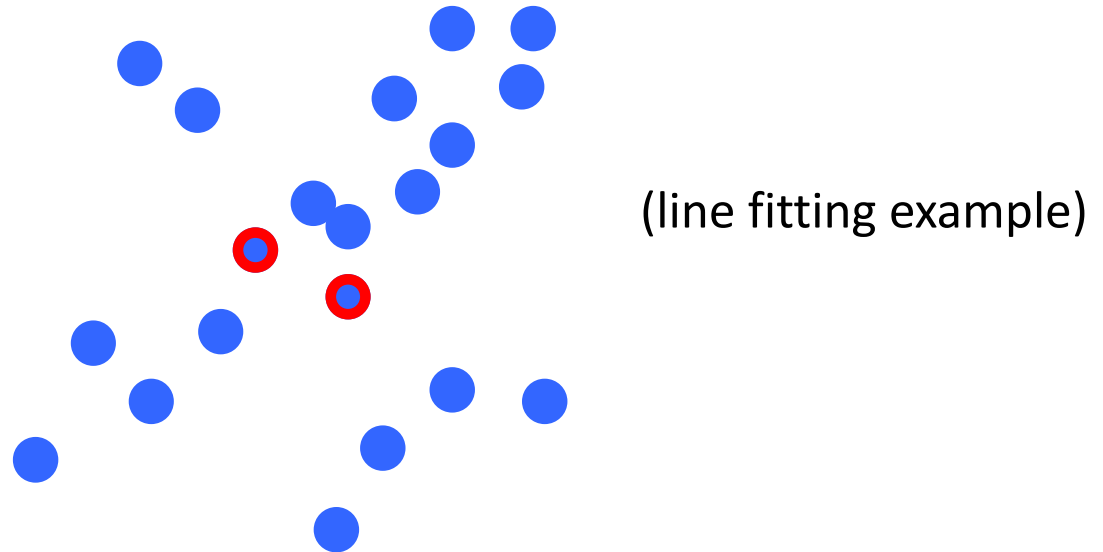
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting and Alignment

- RANSAC



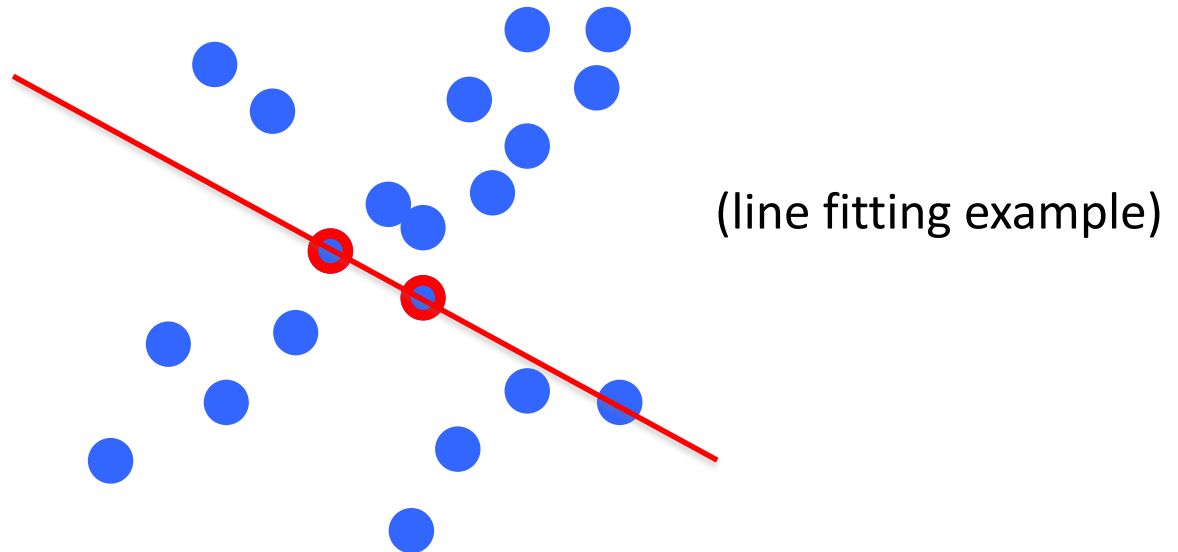
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting and Alignment

- RANSAC

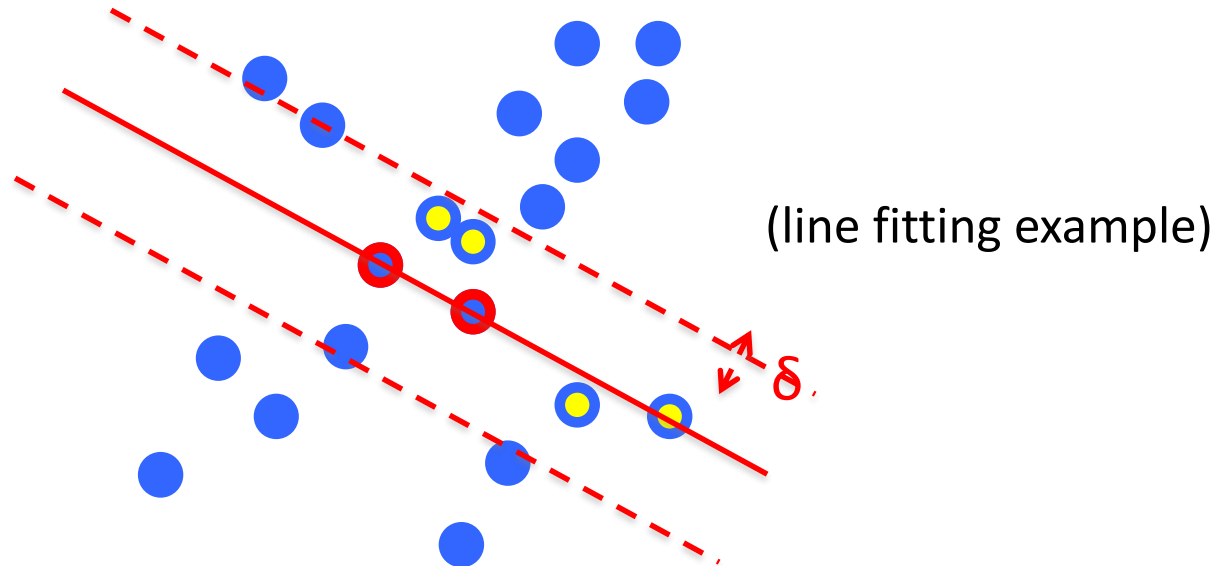


Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
 2. **Solve** for model parameters using samples
 3. **Score** by the fraction of inliers within a preset threshold of the model
- Repeat** 1-3 until the best model is found with high confidence

Fitting and Alignment

- RANSAC



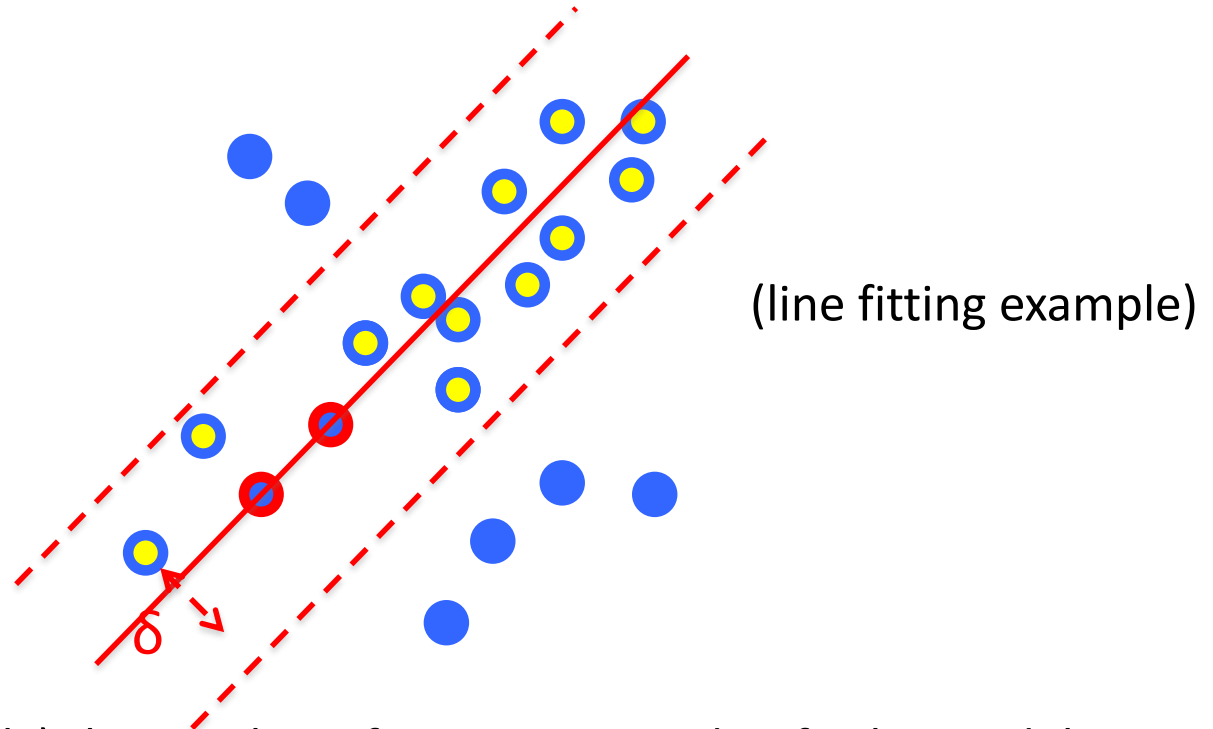
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting and Alignment

- RANSAC



Algorithm:

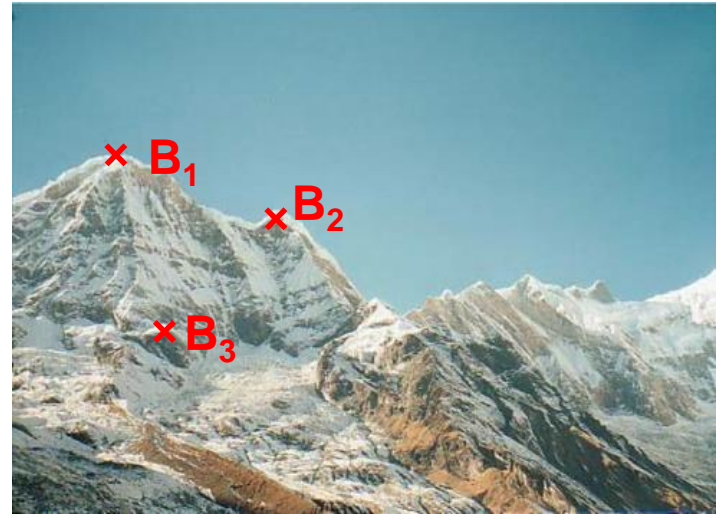
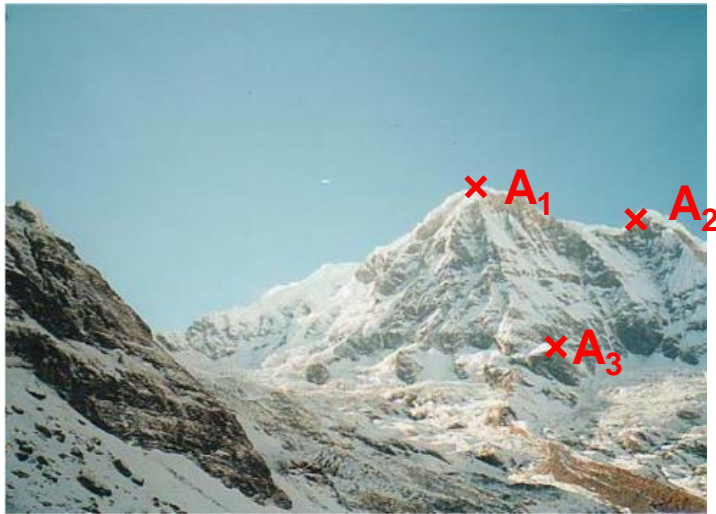
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting and Alignment

- Given matched points A and B, estimate the translation

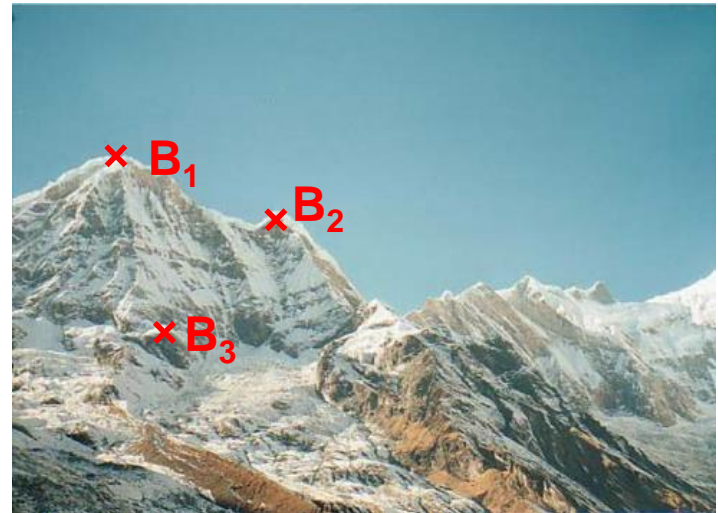
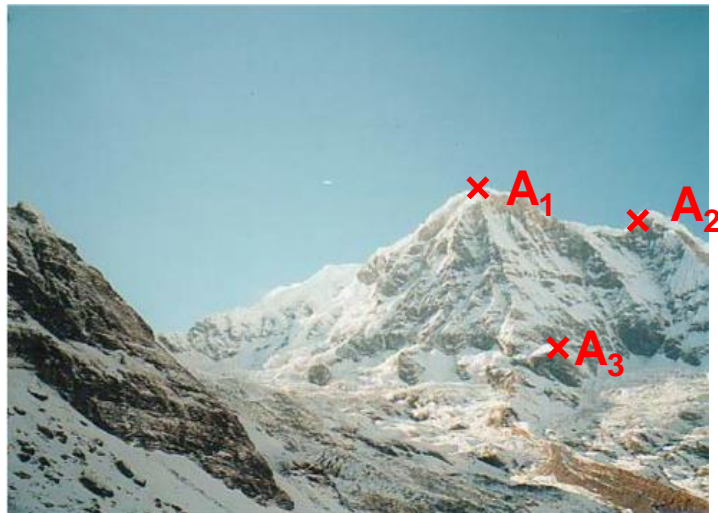
$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



Alignment by Least Squares

1. Write down the objective function
2. Obtain the analytical solution
 - a) Compute derivative
 - b) Compute solution
3. Obtain computational solution
 - a) Write in form $\mathbf{Ap} = \mathbf{b}$
 - b) Solve using pseudo-inverse

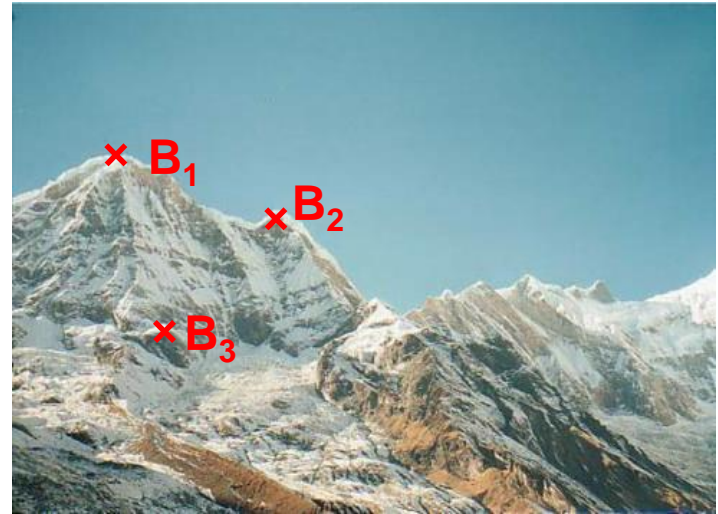
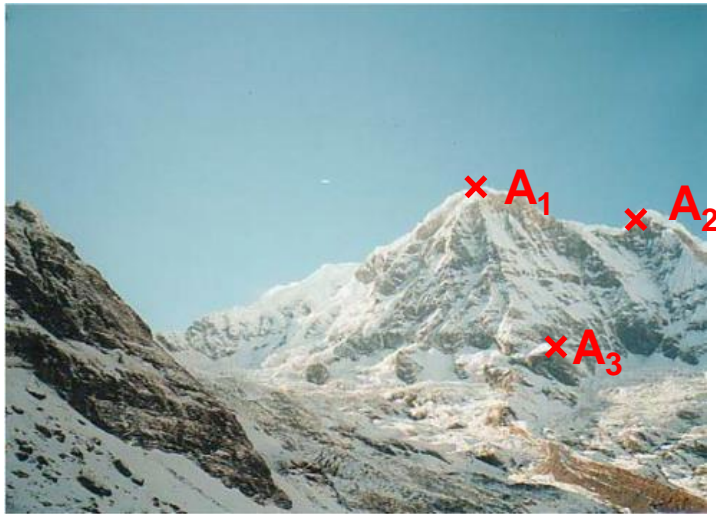
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$



Alignment by RANACAC

1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



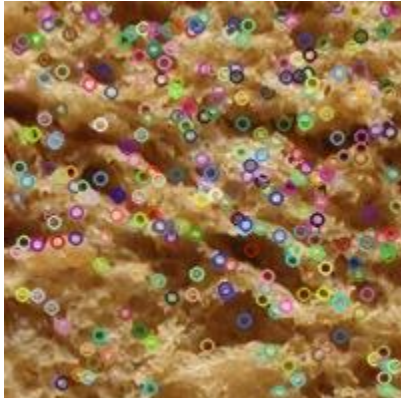
Application Example

- SIFT-based texture classification – how to do this?

bread



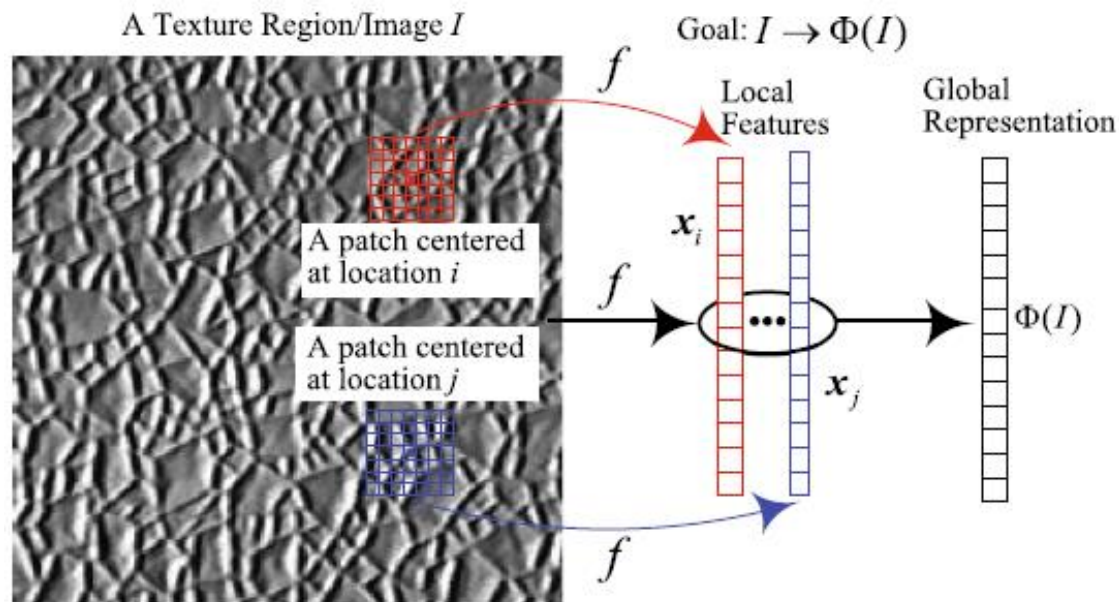
cracker



Problem: the number of SIFT keypoints (and thus the number of SIFT feature descriptors) may vary highly between images

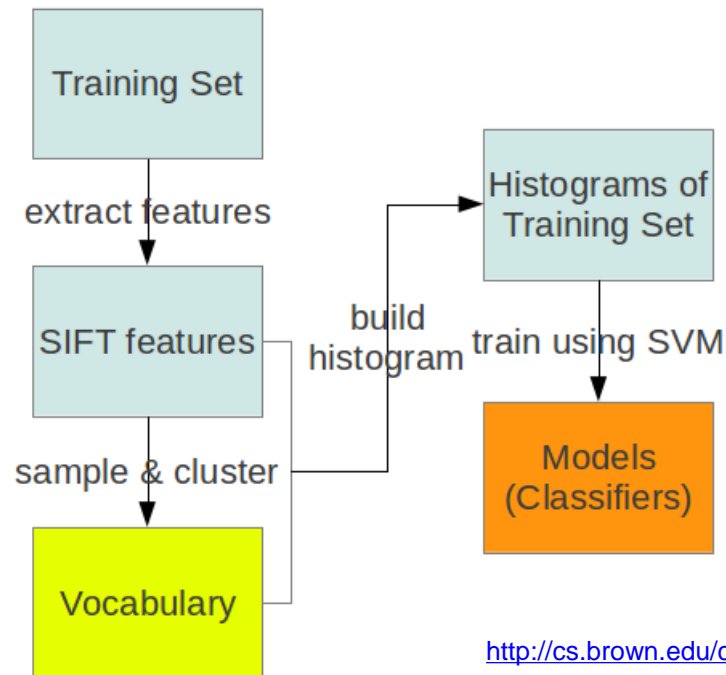
Feature Encoding

- Global encoding of local SIFT features
 - Integrate the local features (SIFT keypoint descriptors) of an image into a global vector to represent the whole image



Feature Encoding

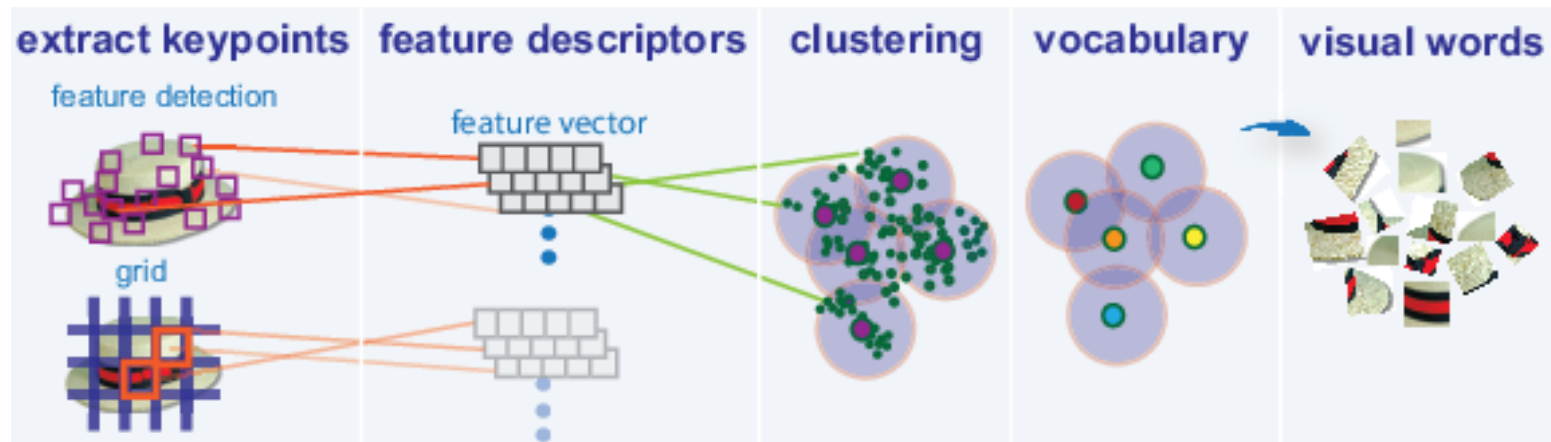
- Most popular method: Bag-of-Words (BoW)
 - The variable number of local image features are encoded into a fixed-dimensional histogram to represent each image



<http://cs.brown.edu/courses/cs143/2011/results/proj3/hangsu/>

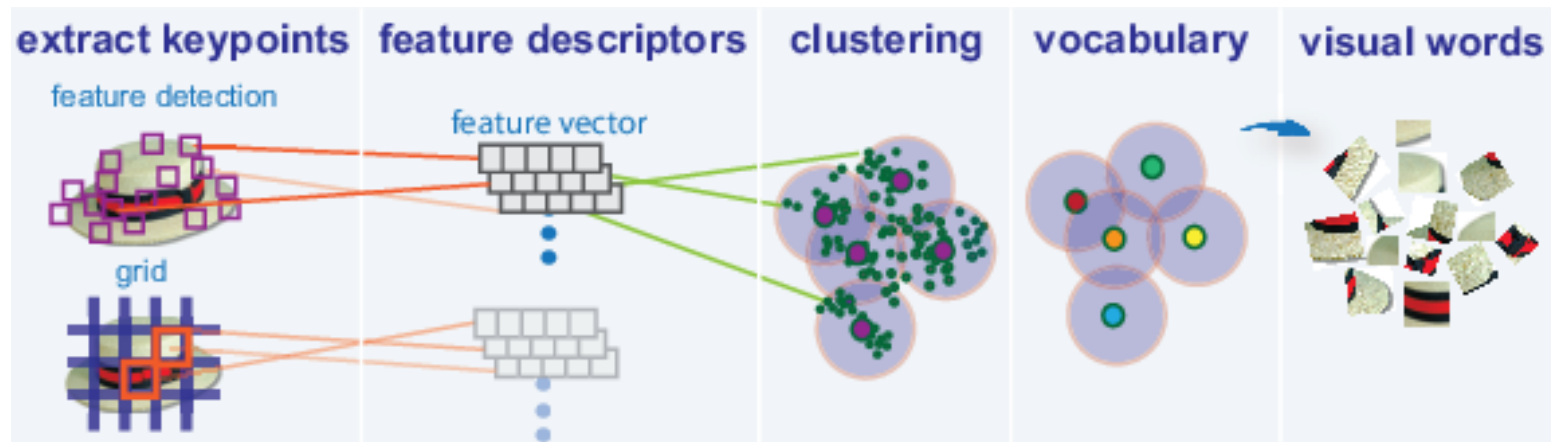
Feature Encoding

- Bag-of-Words (BoW) – step 1
 - Create the vocabulary from the set of local descriptors (SIFT keypoint descriptors) extracted from the training data
 - This vocabulary represents the categories of local descriptors



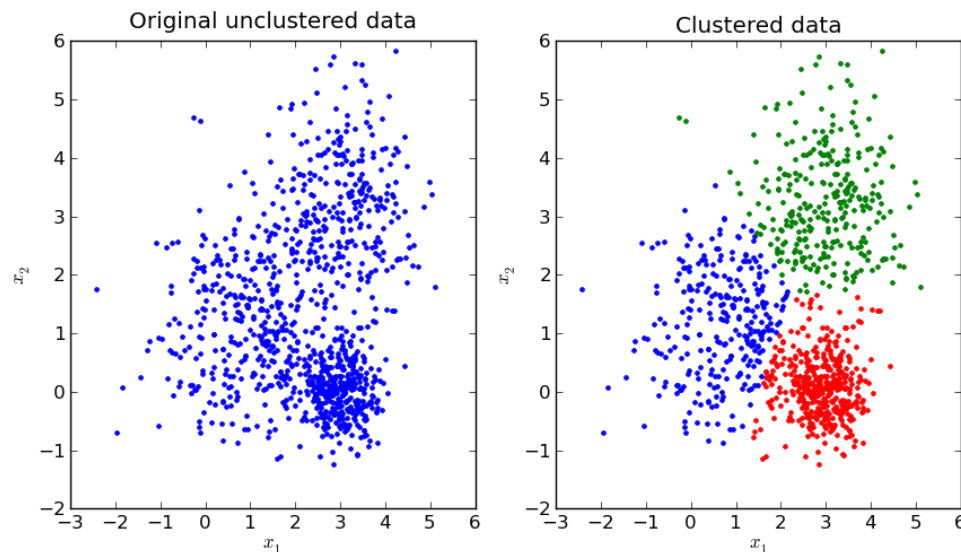
Feature Encoding

- Bag-of-Words (BoW) – step 1
 - Main technique used to create the vocabulary: *k-means clustering*
 - k-means clustering is one of the simplest and most popular unsupervised learning approaches that perform automatic clustering (partitioning) of the training data into multiple categories

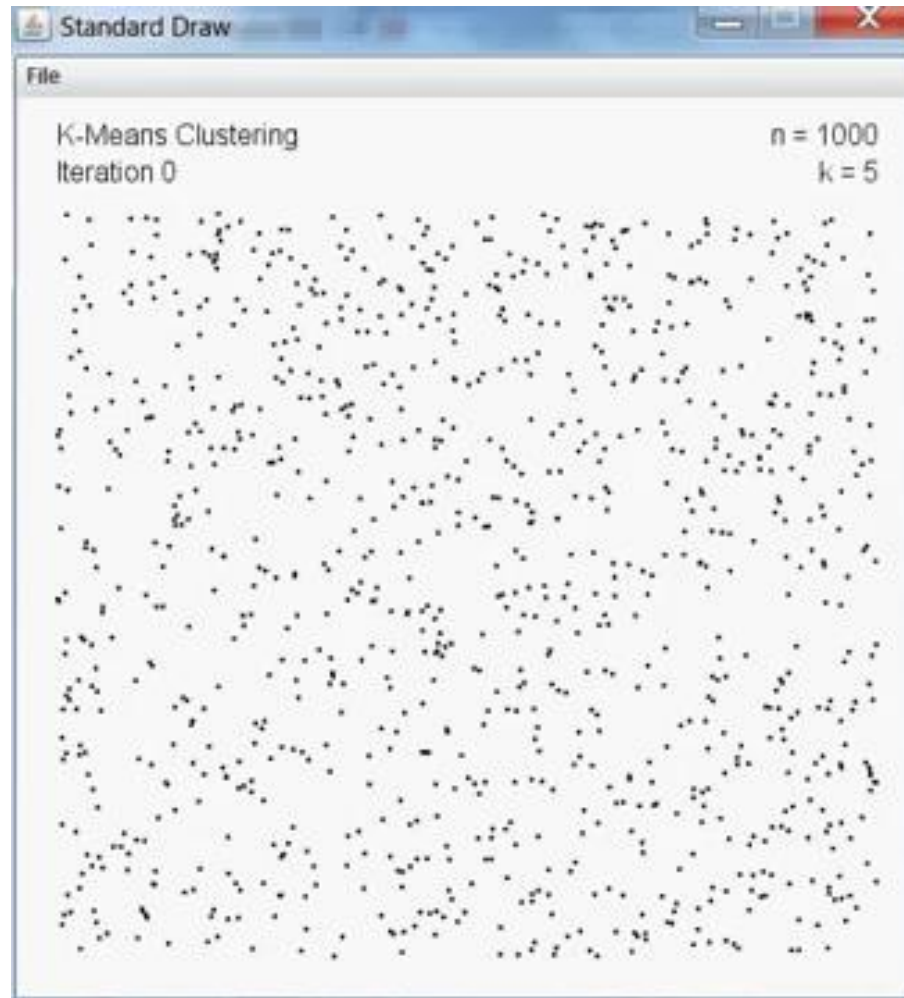


Feature Encoding

- Bag-of-Words (BoW) – step 1
 - K-means clustering:
 - Initialize: k cluster centres, typically randomly
 - Iterate: 1) Assign data (feature vectors) to the closest cluster (Euclidean distance)
2) Update cluster centres as the mean of the data samples in each cluster
 - Terminate: When converged or the number of iterations reaches the maximum



K-Means Clustering



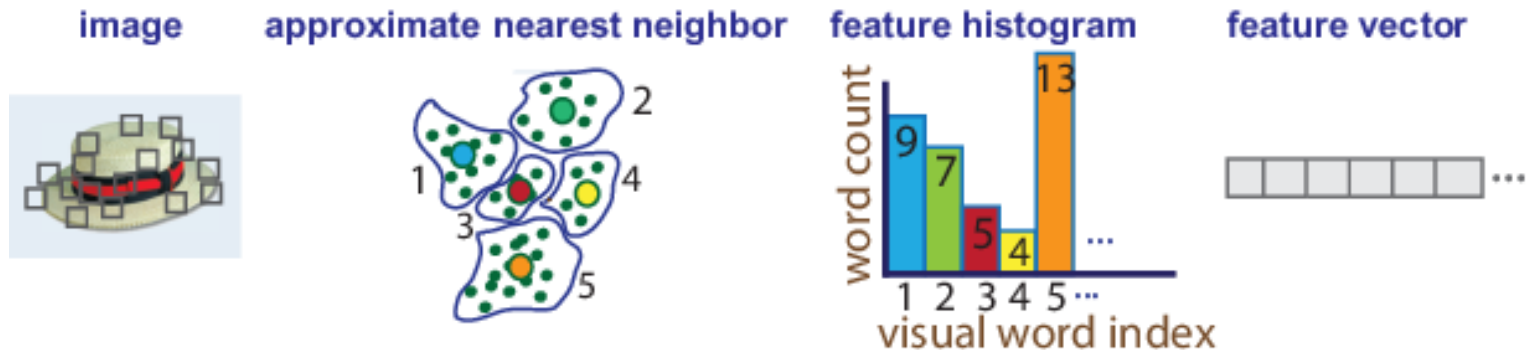
Feature Encoding

- Bag-of-Words (BoW) – step 2
 - The cluster centres are the “visual words” which form the “vocabulary” that is used to represent an image
 - An individual local feature descriptor (e.g. SIFT keypoint descriptor) is assigned to one visual word with the smallest distance



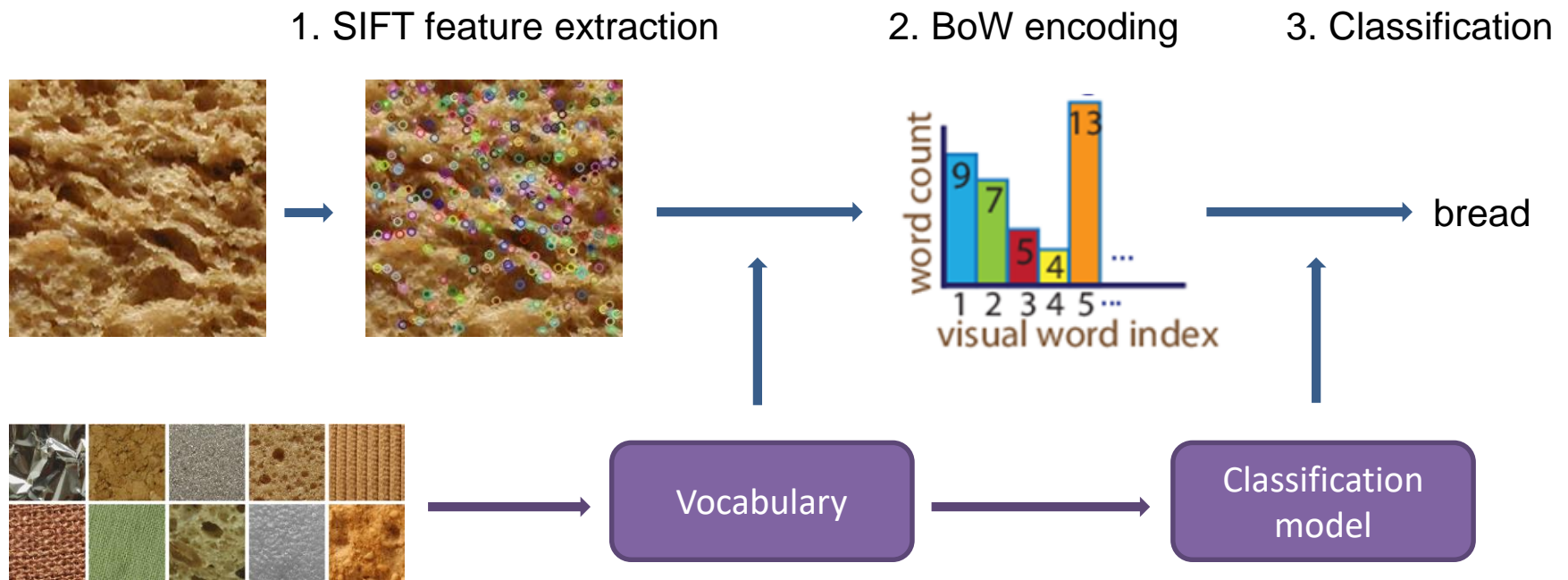
Feature Encoding

- Bag-of-Words (BoW) – step 2
 - For an image the number of local feature descriptors assigned to each visual word is computed
 - The numbers are concatenated into a vector which forms the BoW representation of the image



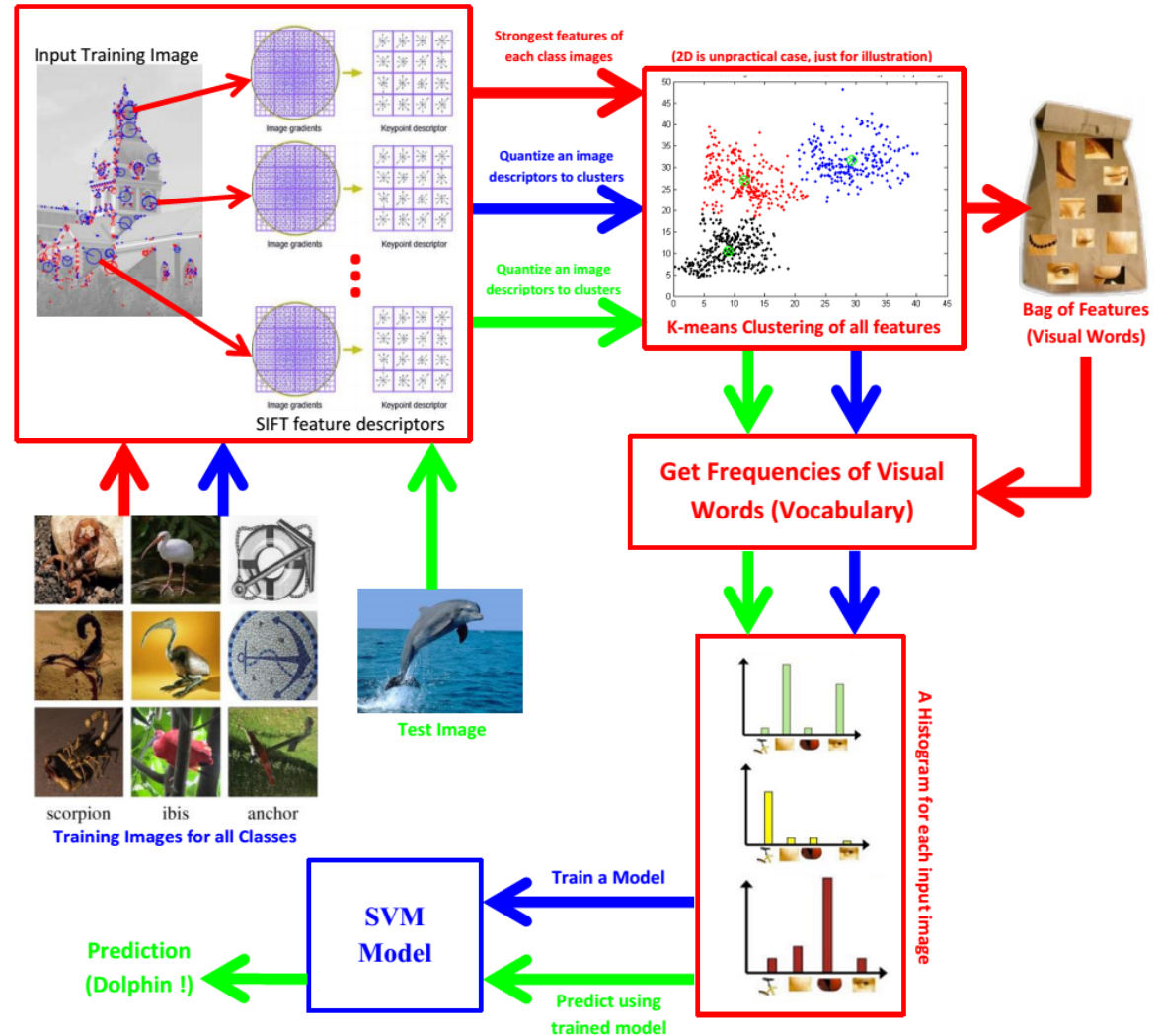
Application Example

- SIFT-based texture classification



Application Example

- SIFT-based texture classification



<http://heraqi.blogspot.com/2017/03/BoW.html>

Feature Encoding

- Local features can be other types of features, not just SIFT
 - LBP, SURF, BRIEF, ORB
- There are also more advanced techniques than BoW
 - VLAD, Fisher Vector
- A very good source of additional information is VLFeat.org
 - <http://www.vlfeat.org/>

Summary

- Feature representation is essential in solving almost all types of computer vision problems
- Most commonly used image features:
 - Colour features (Part 1)
 - Colour moments and histogram
 - Texture features (Part 1)
 - Haralick, LBP, SIFT
 - Shape features (Part 2)
 - Basic, shape context, HOG

Summary

- Other techniques described (Part 1)
 - Descriptor matching
 - Feature encoding (Bag-of-Words)
 - k-means clustering
 - Alignment and RANSAC
 - Spatial transformations
- To be discussed (Part 2)
 - Shape features
 - Shape matching
 - Sliding window detection

References and Acknowledgements

- Szeliski, Chapter 4 (in particular Sections 4.1.1 to 4.1.3 and 4.3.2), Chapter 6 (in particular Sections 6.1.1 to 6.1.4)
- Some content are extracted from the above resource, James Hays slides, and slides from Michael A. Wirth
- L. Liu et al., *From BoW to CNN: two decades of texture representation for texture classification*, International Journal of Computer Vision, 2019
- And other resources as indicated by the hyperlinks